Improving goodput and reliability of ultra-high-speed wireless communication at data link layer level

Von der Fakultät für MINT - Mathematik, Informatik, Physik,
Elektro- und Informationstechnik
der Brandenburgischen Technischen Universität Cottbus-Senftenberg

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Dipl.-Ing.
Łukasz Łopaciński

geboren am 17. Oktober 1985 in Bytów, Polen

Gutachter: Prof. Dr.-Ing. Rolf Kraemer
Gutachter: Prof. Dr.-Ing. Heinrich Theodor Vierhaus
Gutachter: Prof. Dr. Michael Gössel

Tag der mündlichen Prüfung: 8. Februar 2017

Abstract

The design of 100 Gbps wireless networks is a challenging task. A serial Reed-Solomon decoder at the targeted data rate has to operate at ultra-fast clock frequency of 12.5 GHz to fulfill timing constraints of the transmission [1]. Receiving a single Ethernet frame on the physical layer may be faster than accessing DDR3 memory [2]. Moreover, data link layer of wireless systems has to cope with high bit error rate (BER). The BER in wireless communication can be several orders of magnitude higher than in wired systems. For example, the IEEE 802.3ba standard for 100 Gbps Ethernet limits the BER to 1e-12 at the data link layer [3]. On the contrary, the BER of high-speed wireless RF-frontend working in the Terahertz band might be higher than 1e-3 [4]. Performing forward error correction on the state of the art FPGA (field programmable gate arrays) and ASICs requires a highly parallelized approach. Thus, new processing concepts have to be developed for fast wireless communication. Due to the mentioned factors, the data link layer for the wireless 100G communication has to be considered as new research, and cannot be adopted from other systems.

This work provides a detailed case study about 100 Gbps data link layer design with the main focus on communication reliability improvements for ultra-high-speed wireless communication. Firstly, constraints of available hardware platforms are identified (memory capacity, memory access time, and logic area). Later, simulation of popular techniques used for data link layer optimizations are presented (frame fragmentation, frames aggregation, forward error correction, acknowledge frame compression, hybrid automatic repeat request, link adaptation, selective fragment retransmission). After that, data link layer FPGA accelerator processing ~116 Gbps of user data is presented. At the end, ASIC synthesis is considered and detailed statistics of consumed energy per bit are introduced. The research includes link adaptation techniques, which optimize goodput and consumed energy according to the channel BER. To the author's best knowledge, it is the first published data link layer implementation dedicated for 100 Gbps wireless communication shown anywhere in the world.

Kurzfassung

Das Entwerfen von drahtlosen 100 Gbps Netzwerken ist eine herausfordernde Aufgabe. Ein serieller Reed-Solomon-Decodierer für die angestrebte Datenrate muss mit einer ultra hohen Taktfrequenz von 12,5 GHz arbeiten, um die Zeitbegrenzungen der Übertragung zu erfüllen [1]. Das Empfangen eines einzelnen Ethernet Frames auf der physischen Ebene kann schneller ablaufen, als der Zugriff auf den DDR3 Speicher [2]. Darüber hinaus muss der Data-Link-Layer der drahtlosen Systeme mit einer hohen Bitfehlerrate (BER) arbeiten. Die BER in der drahtlosen Kommunikation kann um mehrere Größenordnungen höher liegen, als in drahtgebundener Kommunikation. Der IEEE 802.3ba Standard für 100 Gbps Ethernet, zum Beispiel, limitiert die BER auf 1e-12 auf dem Data-Link-Layer [3]. Die BER von drahtlosen Hochgeschwindigkeits-RF-Frontends, die im Terahertz-Band arbeiten, kann hingegen höher sein, als 1e-3 [4]. Um Forward-Error-Correction auf aktuellsten FPGA zu betreiben, benötigt man einen höchst parallelisierten Ansatz. Daher müssen neue Verarbeitungskonzepte für schnelle drahtlose Kommunikation entwickelt werden. Aufgrund dieser genannten Fakten, und da er auch nicht von anderen Systemen übernommen werden kann, sollte der Data-Link-Layer für die drahtloses 100G Kommunikation als neue Forschung in Betracht gezogen werden.
Diese Dissertation liefert eine detaillierte Fallstudie über ein 100 Gbps Data-Link-Layer Design, wobei der Hauptfokus auf der Verbesserung der Zuverlässigkeit für drahtlose Ultra-Hochgeschwindigkeits-Kommunikation liegt. Zuerst werden die Beschränkungen der verfügbaren Hardware-Plattformen identifiziert (Speicherkapazität, Speicherzugriffszeit und die Anzahl logischer Zellen). Später werden bekannte Verfahren für die Data-Link-Optimierung vorgestellt. Danach werden Simulationen der populären Techniken für den Data-Link-Layer vorgestellt. Außerdem wird ein FPGA Beschleuniger gezeigt, welcher auf dem Data-Link-Layer 116 Gbps an Benutzerdaten verarbeitet. Am Ende wird die ASIC-Synthese betrachtet und eine detaillierte Statistik der verbrauchten Energie gezeigt. Diese Forschung umfasst Verbindungs-Anpassungstechniken, welche den Durchsatz und die verbrauchte Energie optimieren.

Glossary

ACK, ack	Acknowledge
ADC	Analog to digital converter
ARQ	Automatic repeat request
ASIC	Application-specific integrated circuit
AWGN	Additive white Gaussian noise
BB	Baseband
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit error rate
CN	Check node
CRC	Cyclic redundancy check
DEMUX	Demultiplexer
DFG	German Research Foundation
DLL	Data link layer
DMA	Direct memory access
DSSS	Direct Sequence Spread Spectrum
EIRP	Equivalent isotropically radiated power
FEC	Forward error correction
FF	Flip-flop
FIFO	First in first out memory
FMC	FPGA Mezzanine Card
FMC-HPC	FPGA Mezzanine Card - High Pin Count (HPC)
FPGA	Field-programmable gate array
FSM	Finite-state machine
GF	Galois field
GTX/GTH	FPGA high speed serial transceiver
HARQ	Hybrid automatic repeat request
HD	Hard decision decoding
HD-LDPC	Hard decision low-density parity-check
HW	Hardware
IO	Input-output
IRS	Interleaved Reed-Solomon
LDPC	Low-density parity-check
LLC	Logical link control
LUT	Look-up-table
MTU	Maximum transmission unit
MUX	Multiplexer
NIC	Network interface card
PAM	Pulse-amplitude modulation
PCB	Printed circuit board
PCIe	Peripheral Component Interconnect Express
PHY	Physical layer

PLL	Phase-locked loop
PSSS	Parallel Sequence Spread Spectrum
RAM	Random-access memory
RD	Read operation
REQ, req	Request
RF	Radio frequency
RS	Reed-Solomon
RX	Receiver, receiving
SD	Soft decision
SD-LDPC	Soft decision Low-density parity-check
SFP/SFP+	Small form-factor pluggable (Fiber-optic connector)
SMA	Sub-Miniature version *'A'* connector (coaxial RF connector)
SNR	Signal to noise ratio
TPC	Turbo product codes
TX	Transmitter, transmitting
VN	Variable node
WR	Write operation
XST	Xilinx Synthesis Technology (FPGA synthesis tool)

Table of contents

1. Introduction

The ability to communicate without cables has revolutionized the world. One of the first use cases for wireless communication was providing communication with ships by Marconi's radio in 1897 [5]. Since then, wireless communication has been significantly improved and popularized all over the world. Today's wireless transceivers are not only much more robust but also much less expensive. Nowadays, GPS receivers integrated in almost every phone are able to receive signals from satellites that are thousands of kilometers away. Additionally, LTE enabled high-speed wireless Internet and provides communication with goodput[1] of tens of Mbps. This became possible due to challenging research in production technology and communication protocols design. Radio communication has changed our life and every year devices are employed in new applications. Moreover, radio transceivers achieve higher data rate, and wireless communication at 100 Gbps is becoming reality very soon.

At a first glance, high-speed-wireless communication requires only a very fast RF-transmitter and RF-receiver. However, if deeply investigated, several additional issues have to be solved. Employed protocols have to provide mechanisms to control correct transfer of data. This is especially important if a wireless medium is considered. Thus, the receiving device has to inform the transmitter if the data was successfully decoded. Additionally, devices have to deal with unpredictable channel behavior. In case of recurring retransmission of data frames, the goodput of the system falls rapidly and communication latency becomes very high. The protocol has to detect such situations and adaptively react to the instantaneous channel quality. In such situations, the frame size and structure can adaptively be changed. Additionally, the transmitter can include some redundancy bits, so the receiver can fix bit errors in the received data. This is only possible, if devices work in a closed feedback loop and continually exchange information about the link quality. Therefore, the protocol has to control TX and RX windows on both sides and take care of the RF-frontend switching. If a 100 Gbps network were considered, then all these tasks have to be performed in nanoseconds. In this work, all mentioned aspects are discussed, and a prototype of data link layer accelerator for 100 Gbps wireless communication is proposed. The accelerator controls communication reliability and performs tasks that improve link robustness. Operation of the device is fully autonomic and transparent for higher layers.

[1] Goodput is the application-level throughput (i.e., the number of useful information bits delivered by the network to a certain destination per unit of time).

1.1 Introduction to wireless systems

This subchapter introduces some of the most important aspects of wireless communication as well as elements required for common wireless systems.

1.1.1 RF-frontend, baseband, and data link layer processing

Figure 1 presents a typical architecture of a wireless system.

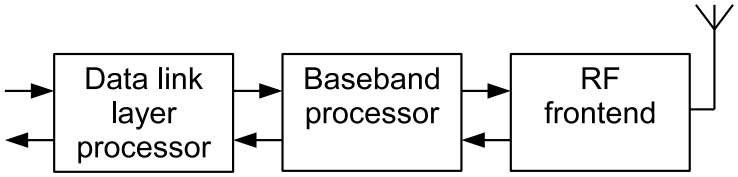


Figure 1: Architecture of a typical wireless system.

Every wireless transceiver has to be equipped with analog frontend that is responsible for filtering, amplifying, and up/down-converting of the RF-signals. Briefly speaking, the frontend consists of all analog elements between the antenna and the mixer (including the mixer).
The baseband processor can be realized as a digital or mixed signal-processor. The most important function of the baseband is recovering data bits and data clock from signals provided by the RF frontend (clock recovery). Additionally, the processor is responsible for synchronization, channel estimation, channel equalization, data scrambling, and managing the RF-frontend.
Data link layer determines access to the medium and controls a logical link between the devices. This may include error detection in a frame, error correction (FEC, channel coding), retransmission of defected frames (ARQ), flow control, and collision avoidance. Briefly speaking, the data link layer is responsible for communication robustness and makes sure that data is transferred between adjacent network nodes without bit errors.

1.1.2 Full-duplex and half-duplex communication

Most radio-transceivers available on the market support half-duplex communication only. This means, that these radios can be only in TX or RX mode, but never in both modes at the same time. In short, the radio-transceivers cannot receive and send data at the same time. Thus, if the transmitting device needs to receive anything, then the TX mode has to be disabled, radio hardware has to be switched to the RX mode, and only then a frame can be received. Additionally, these devices require a protocol that controls which of them is transmitting and for how long. This complicates the data link layer protocols significantly. There are some possibilities to manufacture full-duplex radios, but in case of 100 Gbps transceivers it is too complex and too

expensive in terms of required bandwidth. Although, the design of such radios might be possible in the future.

1.1.3 Return channel and acknowledge messages (ACKs)

The acknowledge messages (ACKs) are transmitted from the receiver device to the transmitter to inform whether the data was successfully received or has to be retransmitted (Figure 2).

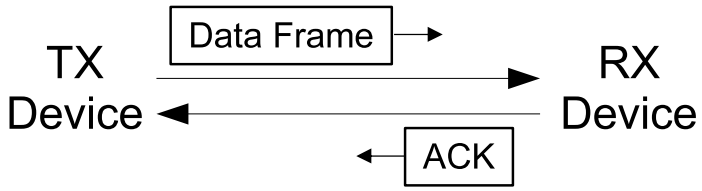


Figure 2: Acknowledge (ACK) messages. ACKs are used to inform the transmitter if data was successfully decoded at the receiver.

There are at least four problems caused by the ACK-messages. Firstly, the ACK message is not carrying any user data, and the data transmission is paused during ACK exchange. Secondly, the ACK-frame can be lost and state machines on both sides have to deal with this problem. The simplest solution is to start timers, and if ACK does not arrive in the required time, the transmission is restarted. This additionally reduces goodput, because the devices have to wait for a timeout. Thirdly, to send the ACK-frame, both devices (transmitter and receiver) have to switch radios from TX to RX and from RX to TX respectively. This costs some additional time, and goodput is reduced again. Fourth, both devices have to be equipped with fully functional transmitter and receiver, even if data transmission is unidirectional. This doubles the resources required for manufacturing those devices. The return channel may be also used to exchange some communication settings, link quality information, and flow control mechanisms. Thus, the return channel is necessary for most protocols, even if user data transmission is unidirectional only. If the return channel is considered for half-duplex systems, then special care has to be taken during protocol design. If ACKs and other messages are sent too often and/or are too large, then goodput is significantly reduced.

1.1.4 Radio turnaround time

Switching between RX and TX modes can be very expensive in terms of time. RF-transceivers usually require some time to switch from RX to TX mode and vice versa

(Figure 3). During the switching, no data is transferred and effective data goodput is reduced[2].

Figure 3: Explanation of RX-turnaround time. Radio transceivers require some time to switch between the TX and RX modes (and vice versa). During this time, data cannot be exchanged.

To achieve the highest user data goodput, the protocol has to reduce the number of RF-turnarounds (RF-switches) per second. This is not always easy to realize in practical implementations due to automatic repeat request (ARQ) memory buffers. This problem is explained further in details.
The RF-turnaround time can vary from device to device. For example, state of the art, low-cost CC1101 (Texas Instruments) requires up to ~800 us for mode switching [6]. In case of high-performance radios, the switching time is optimized and significantly shorted. For 802.11ad RF-frontends (WLAN operating in the 60 GHz band), the time is standardized to be less than 1 us [7]. The best TX-RX switching performance is achieved, if the transmitting and receiving circuits use separate analog elements. Then only an antenna switch is required to change the mode. The TX and RX hardware elements are active all the time, and no settling-time[3] [6] is required to turnaround the mode (Figure 4).

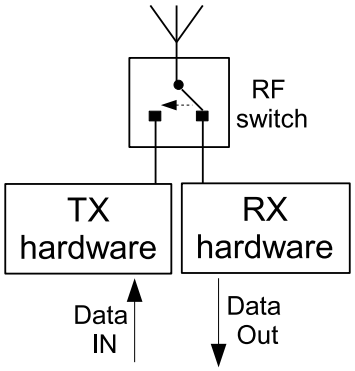


Figure 4: Optimized radio architecture for ultra-fast turnaround time. The radio is using two independent hardware circuits for receiving (RX) and transmitting (TX). The mode is selected by changing the state of the RF switch.

[2] Half-duplex communication is considered. Full-duplex RF-transceivers are out of the scope of this work due to self-interference issues.
[3] Settling-time is the time it takes for a RF-transceiver to settle to achieve the specified operating mode (e.g., RX or TX mode).

1.1.5 PHY-preambles

Every frame sent by a wireless transmitter is extended by a PHY-preamble (Figure 5). The preamble is a pattern of defined symbols and is transmitted before the data, so the receiver can adjust the RF-fronted and baseband parameters (e.g., power amplifier gain, center frequency, clock recovery circuit). The sequence is known to the receiver, and therefore the hardware settings can be adjusted before user data is received. In more advanced communication systems, the preamble can be used for channel estimation that is a part of channel deconvolution process [8]. In such case, the transmitted signal is recovered from the received signal that is convolved with impulse response of a communication channel.
The preamble is an important part of a frame, but during preamble transmission, user data is not exchanged. Thus, preambles are reducing the effective goodput of communication systems. To mitigate this effect, frame aggregation techniques can be employed.

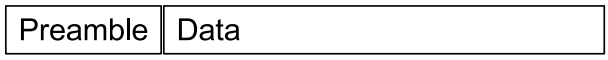


Figure 5: Data frame with a preamble.

1.1.6 Forward error correction (channel coding)

Forward error correction (FEC, channel coding) adds extra redundancy bits to improve reliability of the user data transmission. The receiver uses the bits to localize and correct errors caused by transmission impairments. The error correction performance mainly depends on the number of extra bits and complexity of the decoding algorithm. If more bits are added, or the algorithm is more complicated, then more errors can be detected and corrected in a received data stream. FEC is a powerful technique to improve transmission reliability, but the redundancy bits reduce the effective user data goodput. Additionally, the most powerful correction algorithms are complex and require high computing power.

1.1.7 Goodput and overall transmission efficiency

Goodput is defined by the number of useful information bits delivered by the network to a certain destination per unit of time [9]. The goodput is lower than the throughput. The throughput is the gross bit rate that is transferred physically. Thus, RF-turnaround time, channel coding, defected frames retransmission, transmission of ACKs, and preambles reduce the goodput seen from the user point of view. Moreover, mentioned factors lower energy efficiency of the system, and lead to shorter operating time on a battery for mobile devices. Thus, increasing the overall system efficiency (goodput, energy efficiency per bit) is one of the goals of this work.

1.1.8 Parallel sequence spread spectrum (PSSS)

PSSS is one of spreading techniques used for improving communication robustness at physical layer (PHY). Figure 6 depicts the idea of operation of a PSSS spreading employed in the targeted 100 Gbps physical layer (developed under Real100.COM project[4] [10]). Each bit is multiplied by a cyclically shifted spreading code (usually an m-sequence or a Barker code [11] is used). After that, all values are added together and a single multilevel PSSS symbol is formed. In the targeted 100 Gbps transmitter developed under Real100.COM project, a single PSSS symbol consists of 15 chips, contains 13-15 data bits, and up to 2 cyclic prefixes [12]. Thus, the system requires a spreading code of length of at least 15 chips. The main advantages of the PSSS are improved communication robustness and computation architecture that can be implemented in an analog circuit. This allows to avoid analog to digital converters (ADCs) for baseband implementation on the receiver side. Design of an ADC for the targeted 100 Gbps transmission is difficult. Therefore, the PSSS reduces complexity of the targeted transceiver hardware. More details can be found in [12]–[15].

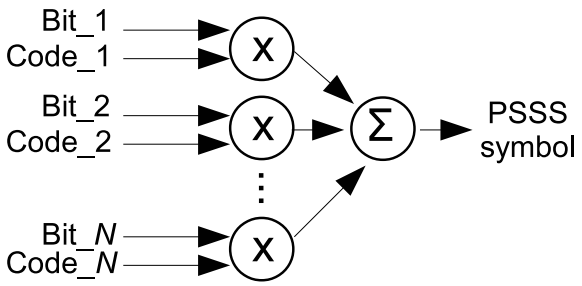


Figure 6: PSSS spreading circuit employed in the targeted 100 Gbps baseband (Real100.COM project).

1.2 Progress in designing 100 Gbps RF-transceivers

Within the last three years, a few new approaches for 100 Gbps wireless communication have been proposed. Research on physical transceivers and baseband processing changed the state of the art in the targeted area. Design blocks required to modulate 100 Gbps wireless signal in the Terahertz band are close to release for experimental setups. In [16] a 100 Gbps baseband signal has been sent over a 237.5 GHz link. Similar results are shown in [17], and [13]. More THz communication activity on the physical layer is documented in [18], [19], [20]. Table 1 summarizes reported transmission experiments in the Terahertz band [4].

[4] See section 3.1 to get an overview of the complete system as investigated in the DFG-SPP1655.

From the data link layer point of view, research on power effective error control mechanisms presented in [21] is especially interesting. The authors consider a hybrid-ARQ approach for nanonetworks operating in 300 GHz band with OOK (On/Off Keying) modulation. The presented simulation models uses Hamming(15,11) channel coding with ARQ (Automatic Repeat Request). This uncomplicated solution is considered due to millimeter distances used in the targeted application, and is not a recommended option for general purpose 100 Gbps transceivers due to poor error correction efficiency[5]. However, proposed power estimation techniques and the mathematical way of designing a power efficient data link layer can be used as a starting point for future investigations.

Frequency [GHz]	Data rate [Gbps]	BER / EVM	Distance [m]	Reference
300	24	BER < 1e-10	0.5	[22]
120	10	BER < 1e-10	200	[23]
87.5	100	BER ≈ 1e-3	1.2	[24]
237.5	100	BER ≈ 3.4e-3	20	[25]
240	30	EVM < 16%	40	[26]
220	30	BER ≈ 1e-8	20	[27]
300	24	BER < 1e-9	0.3	[28]
300	48	BER < 1e-10	1	[29]
237.5	100	BER ≈ 1e-3	20	[30]
100	100	BER < 3.8e-3	0.7	[31]
400	40	BER ≈ 1e-3	2	[4]

Table 1. Summary of reported transmission experiments in the Terahertz band (EVM – error vector magnitude; BER — bit error rate; source: [4]*).*

1.3 Motivation and research objectives

Future applications pose tough challenges on wireless systems and therefore are a big driver for new research directions. For example, video applications based on the planned Super Hi-Vision standard require data rates up to 72 Gbps and supports a stream with resolution of 7680 x 4320 pixels at 60 frames per second [32]. Clearly, none of the state of the art wireless systems can support such extremely high data rates. The fastest wireless technology available, based on wireless LAN 802.11ac (5 GHz) and 802.11ad (60 GHz), achieves data rates of "only" 7 Gbps [33].
To perform the 100 Gbps transmission, not only fast physical layer (PHY) is required. This work focuses on the overall transmission goodput and reduction of the overall overhead induced by the data link layer protocols for ultra-high-speed

[5] Length of Hamming(15,11) code word is too short and the redundancy data is used inefficiently. This is investigated in section 3.15.3, and is proved by simulations shown in Figure 121. Moreover, the proposed Hamming decoding is a simple single pass algorithm and is usually less efficient than iterative solutions proposed in sections 2.4.7 and 3.16.

communication. Because of this work, a hardware accelerator for data link layer is presented. The implementation enables processing of 100 Gbps streams, and is one of the first data link layer (DLL) processors dedicated for 100 Gbps wireless communication. The presented DLL protocol is designed from scratch and optimized for the targeted application and hardware platform (FPGA/ASIC). The frame format, return channel, aggregation, fragmentation, link adaptation, forward error correction, selective fragment retransmission, and hybrid-ARQ schemes are investigated in details. Moreover, the approaches are redesigned to fulfill timing requirements of 100 Gbps networks. The main technical idea is to improve transmission robustness for very-high-speed wireless communications. Thus, investigation of 100 Gbps forward error correction with link adaptation algorithms is one of the key aspects of this work. All investigated solutions are validated on the VC709 Virtex7 board. Thus, the proposed schemes are tested against consumed hardware resources and operational clock frequency. Additionally, the implementation is synthesized into the IHP 130 nm technology and consumed energy per processed data bit is investigated.

1.4 Structure of the thesis

First chapter (“Introduction”) introduces the reader to the wireless communication and discusses the main challenges of the 100 Gbps data link layer processor design.

Second chapter (“State of the art in improving communication goodput and reliability”) introduces and explains mechanisms that are used to improve reliability and goodput of communication protocols. All of the presented methods are used in common wireless standards (e.g., WLAN, GSM, LTE). Additionally, theory of operation and simulation results of the methods are discussed in details.

Third chapter (“Searching optimal architecture for 100 Gbps data link layer processor”) presents simulation preconditions, simulation models, and simulation results of the proposed DLL scheme. Firstly, the frame length, fragmentation threshold, and ACK-frame length are investigated. After that, the research focuses on FEC, Hybrid-ARQ, and link adaptation algorithms. The most important contribution of the third section is an improved coding scheme proposed for turbo product codes (TPC). Moreover, the chapter discusses required resources for hardware accelerated FEC engine.

Fourth chapter (“Results”) gives an overview of the implemented architecture and discusses energy efficiency per processed data bit. All power and energy related aspects are discussed according to 130 and 40 nm CMOS technologies. The most important contribution of the last section is definition of a tradeoff between ARQ and FEC according to the energy efficiency.

The last chapter (“Appendix”) explains the implemented demonstrator and the main issues of the implementation.

1.5 Publications list

The research conducted in the PhD project was driven by the need to design and implement a data link layer protocol for 100 Gbps transceivers operating in the Terahertz band. During the course, two journal articles and nine conference papers have been published. Two more articles are currently in a review process. In total, three journals and ten per reviewed conference articles have been prepared. The next subsection summarizes the articles.

1.5.1 Journal articles

Journal article #1:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. (2016). „100 Gbps data link layer – from a simulation to FPGA Implementation", in *Journal of Telecommunications and Information Technology (JTIT),* ISSN Online: 1899-8852; ISSN Print: 1509-4553;

In [34] a case study of a simulation and hardware implementation of a data link layer for 100 Gbps Terahertz wireless communication is presented. The following aspects are introduced: an acknowledge frame compression, frame fragmentation and aggregation, Reed-Solomon forward error correction, and an algorithm to control transmitted data redundancy. The most important conclusion is that changing the frame fragmenting size influences mainly uncoded transmissions. Thus, when FEC is applied, the fragment size can be set to a constant value. Moreover, memory footprint can be significantly reduced if HARQ type II is replaced by the type-I with the proposed link adaptation algorithm.

Journal article #2:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. "Data Link Layer Considerations for Future 100 Gbps Terahertz Band Transceivers", in *Journal of Wireless Communications and Mobile Computing,* ISSN Online: 1530-8677; ISSN Print: 1530-8669;

Processing 100 Gbps data streams on a state of the art FPGA requires a highly parallelized approach. Firstly, hardware constraints of available hardware platforms are investigated (memory capacity, memory access time, processing effort, required chip area). Later, simulations of popular techniques used for data link layer optimizations are presented (frames fragmentation, frames aggregation, forward error correction, acknowledge frame compression, hybrid automatic repeat request). At the end, a fully functional data link layer FPGA demonstrator is presented.
Journal article #3:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. "Improving energy efficiency using a link adaptation algorithm dedicated for 100

Gbps wireless communication", in review *AEÜ - International Journal of Electronics and Communications,* ISSN: 1434-8411

This paper presents a link adaptation algorithm dedicated for 100 Gbps wireless transmission. Interleaved Reed-Solomon codes are selected as forward error correction algorithms. The redundancy of the codes is selected according to the channel bit error rate. The uncomplicated FEC scheme allows implementing a complete data link layer processor in an FPGA. The proposed FPGA-processor achieves 169 Gbps throughput. Moreover, the implementation is synthesized into 40 nm CMOS technology and the described link adaptation algorithm allows reducing consumed energy per bit to values below 1 pJ/bit at BER < 1e-4. With higher BER, the energy increases up to ~13 pJ/bit.

1.5.2 Peer reviewed conference papers

Conference paper #1:
Lopacinski L., Brzozowski M., Kraemer R., Nolte J. (2014), "100 Gbps Wireless - Challenges to the Data Link Layer", in Proc. *IEICE Information and Communication Technology Forum (IEICE ICTF)*, Poznan, Poland.

In this paper [35], basic problems of an implementation of a parallel data link layer processor are discussed. Such a high data rate (100 Gbps) requires a fast speed and low latency memory for automatic repeat request (ARQ). Use of DDR3 memory leads to too long latencies, while use of FPGA on chip block RAMs requires wide data buses and has the problem that the memory size is limited. Moreover, FEC algorithms have to be chosen very carefully due to complexity issues. A complicated FEC leads to huge hardware structures. Even with less complicated FEC, there is probably a need to use multiple FPGA devices and fast interfaces between them. For this reason, high-speed serial IO transceivers are introduced (GTH/GTX/GTZ).

Conference paper #2:
Lopacinski L., Brzozowski M., and Kraemer R. (2015). "A 100 Gbps data link layer with a frame segmentation and hybrid automatic repeat request". In *Science and Information Conference 2015 (SAI2015)*, London, United Kingdom.

The paper [36] presents Matlab simulation results of the DLL protocol. Frame aggregation, FEC codes, and HARQ schemes are in the scope. Frame fragmentation leads to long ACK-frames, and this issue has to be improved by employing ACK compression approaches. Reed-Solomon codes are selected for the FEC engine because of relative high decoding throughput and sufficient error correction performance comparing to convolutional codes. Moreover, parameters of a physical layer and theirs influence on the system performance are discussed.

Conference paper #3:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. (2015). "100 Gbps Wireless – Data Link Layer VHDL Implementation", in Proc. of the *18th Conference on Reconfigurable Ubiquitous Computing*, Szczecin, Poland.

This paper [1] describes hardware used for 100 Gbps data link layer implementation. So fast stream processing requires a highly parallelized approach. Timing requirements of 100 Gbps networks are so demanding that there is no chance to deal with this task as a single processing stream in an FPGA. Due to this reason, the authors introduce and validate a dedicated lane-architecture that solves the issue. The FPGA lane processing is explained in details, and the most important parameters of the FPGA implementation are introduced.

Conference paper #4:
Lopacinski, L., Nolte, J., Büchner, S., Brzozowski, M., and Kraemer, R. (2015). „Parallel RS error correction structures dedicated for 100 Gbps wireless data link layer". In *15th IEEE International Conference on Ubiquitous Wireless Broadband 2015: Special Session on Wireless Terahertz Communications (IEEE ICUWB 2015 SPS 02)*, Montreal, Canada.

One of the most calculation intensive operations for 100 Gbps wireless frame processing is FEC. Thus, there is a need to find a high-parallelized FEC structure for the targeted Virtex7 device. In the paper [37], interleaved Reed-Solomon (IRS) codes are proposed to reach the 100 Gbps goodput. The main task is to select the best RS coding parameters for the targeted device and expected channel BER.

Conference paper #5:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. (2015). "Design and implementation of an adaptive algorithm for hybrid automatic repeat request". In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (IEEE DDECS2015),* Belgrade, Serbia.

Transmission efficiency is an interesting topic for data link layer developers. The overhead of protocols and coding has to be reduced to a minimum. This is especially important for high-speed networks, where a small degradation of efficiency will degrade the goodput by several Gbps. In the paper [38] a redundancy-balancing algorithm for an adaptive HARQ with RS coding is introduced. Mathematical description, hardware block diagram, and all necessary arithmetic simplifications are explained in details. The algorithm can be represented by basic logical operations in FPGA hardware. Thus, it requires very little hardware resources.

Conference paper #6:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. (2015). A "100 Gbps data link layer with an adaptive algorithm for forward error correction, in Proc". *IEICE Information and Communication Technology Forum (IEICE ICTF)*,

Manchester, United Kingdom.

To achieve the highest user data goodput, the overhead induced by the data link layer protocol has to be reduced to a minimum. It means that the payload has to dominate in the frame, and the frame size has to be increased to at least 4 MB. This approach has advantages on links with a relatively low bit error rate. If the channel quality is low (high bit error rate), then this solution reduces the goodput or will block the link completely. Thus, in the paper [39] a dedicated HARQ approach with selective fragment retransmission is proposed. Additionally, some redundant FEC bits are added to the frame-fragments. To reduce the negative impact of the redundancy bits on the system goodput, the protocol adopts the number of the redundant bits according to the channel quality.

Conference paper #7:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. (2015). "Design and Performance Measurements of an FPGA Accelerator for a 100 Gbps Wireless Data Link Layer", in Proc. *International Symposium on Signals, Systems and Electronics (ISSSE)*, Gran Canaria, Spain.

To achieve 100 Gbps wireless transmission, not only a very fast physical layer is required. The effort of the analog transceiver can be wasted due to the overhead induced by the higher network layers. Delays and latencies caused by duplex switching can dramatically reduce the goodput of the link. In every microsecond of a delay, 12.5 kB of the data transfer is wasted. Therefore, there is a need to extend the frame size, but that leads to a higher packet error rate. To deal with this problem, dedicated frame format is employed. The protocol uses a frame with subframes. The frame is divided into subframes and the subframes can be selectively repeated. This allows to retransmit only a small part of the defect frame. Additionally, the protocol proposed in this paper [40] changes the subframe size to improve communication robustness.

Conference paper #8:
Lopacinski, L., Nolte, J., Buechner, S., Brzozowski, M., and Kraemer, R. (2016). „Improved Turbo Product Coding dedicated for 100 Gbps Wireless Terahertz Communication", in Proc. *IEEE PIRMC 2016*, Valencia, Spain.

In the article, an improved turbo product-decoding scheme is proposed. The new method is almost as effective as hard decodable low-density parity check codes (HD-LDPC). Due to the modified code word shape, no external interleavers are required to correct burst errors. If the decoder uses Reed-Solomon (RS) codes, then error correction performance against burst errors is significantly higher than the gain provided by HD-LDPC with an external interleaver. An additional advantage is a possibility to design a dedicated decoder for Virtex7 FPGA serial transceivers. The targeted platform is Virtex7 FPGA, but the solution can be easily scaled on other technologies.

Conference paper #9:
Lopacinski, L., Buechner, S., Nolte, J., Brzozowski, M., and Kraemer, R. (2016). „Towards 100 Gbps wireless communication: energy efficiency of ARQ, FEC, and RF-frontends", in Proc. *ISWCS 2016,* Poznan, Poland.

The paper introduces recent results of 100 Gbps wireless transceiver design. Furthermore, energy for retransmissions and forward error correction is compared. The presented model estimates energy boundaries, when the fragment selective retransmissions are more energy efficient than forward error correction (FEC). In the targeted system, the FEC is relatively expensive and the FEC mode with the highest goodput is not optimal in terms of consumed energy per bit. Moreover, energy efficiency of the data link layer processor to the energy required to transmit a single bit on the physical layer is compared. In most cases, gain obtained by forward error correction consumes more energy than the gain obtained by power amplifiers in the terahertz band.

Conference paper #10:
Lopacinski, L., Buechner, S., Nolte, J., Brzozowski, M., Krishnegowda, K., and Kraemer, R. (2016). „Towards 100 Gbps wireless communication: investigation of FEC interleavers for PSSS-15 spreading", in review *EUROCON 2017,* Ohrid, Macedonia.

The main aspect considered in this paper is comparison of interleaver sizes for convolutional and low-density parity-check codes (LDPC) employed for 100 Gbps wireless communication at 240 GHz with parallel sequence spread spectrum (PSSS). Interleavers required for PSSS-15 and convolutional codes are larger in silicon area than a complete Reed-Solomon decoder. Thus, convolutional codes are not recommended for the targeted application. LDPC codes require 10× smaller interleavers than convolutional codes, and seems to be a good choice for the targeted data rate. Alternatively, interleaved Reed-Solomon decoders are proposed. Hard decision RS decoding reduces the size of the targeted forward error correction processor and provides error correction performance not lower than hard decision convolutional codes at the same code rate.

2. State of the art in improving communication goodput and reliability

This chapter presents typical tasks performed by data link layer (DLL) processors and focuses on improving goodput and robustness of ultra-high-speed wireless transmissions. Thus, the uppermost sublayer, logical link control (LLC) [41] is discussed in most cases. The most important features are frames aggregation, selective fragment retransmission, and forward error correction. Finding a tradeoff between these approaches for 100 Gbps networks is the key element of this work. The designed demonstrator uses point-to-point communication with statically assigned master-slave roles. Therefore, the second sublayer – media access control (MAC) [42] is not discussed.
At the end of the chapter, examples of modern DLL-processors are introduced, and the dissertation is compared to other published work.

2.1 Frames fragmentation

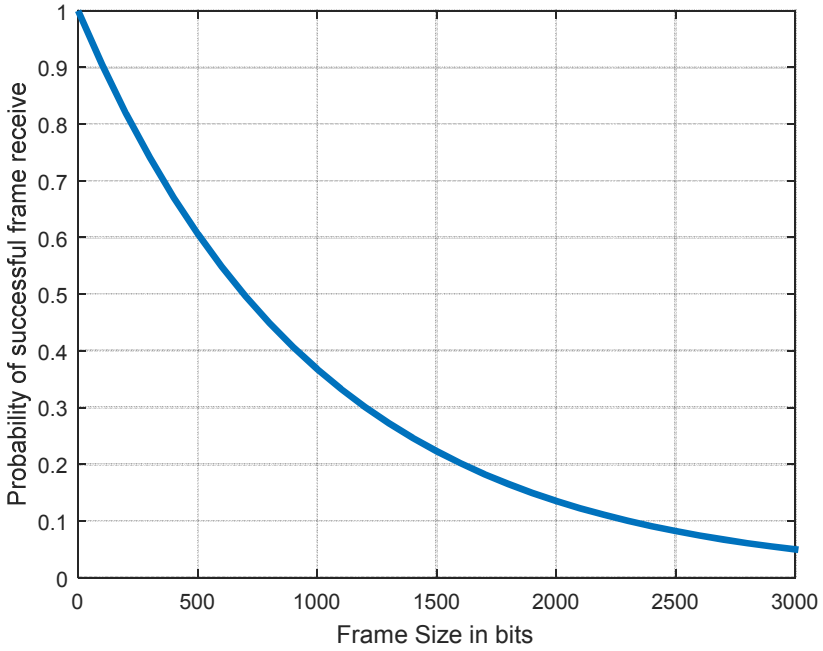


Figure 7: Probability of successful frame reception as a function of frame size at BER = 1e-3. For longer frames, the probability of faulty bits is higher, which reduces the probability of successful frame reception.

Frame length and frame error rate are strongly correlated (Figure 7). For a longer frame, the probability that at least one of the bits will be altered during transmission is higher, due to channel impairments. Fewer bits in a frame reduce the number of possibilities for bit errors to occur. Thus, shorter frames are preferred in a noisy medium. This observation leads to a frame-fragmentation concept. Long frames can be split into several shorter frames [43] (Figure 8). This operation improves frame error rate and data goodput. This is especially important for wireless 100 Gbps implementations, where the frame length has to be maximally extended to achieve high transmission efficiency (goodput) and to reduce idle time of the RF-frontend.

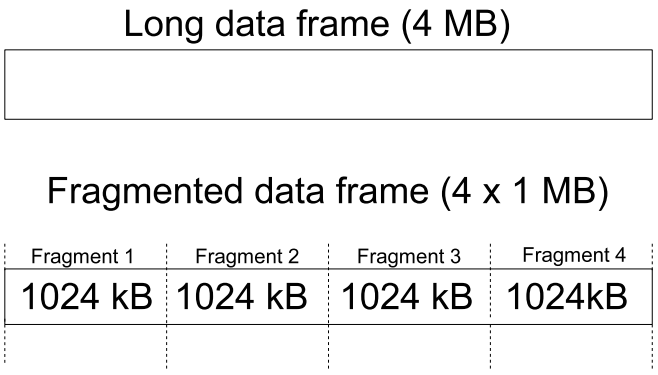


Figure 8: Frames fragmentation concept - long frames are split into shorter frames. Therefore, the frame error rate is reduced.

If the fragmented frame is more robust against BER (Figure 8), successful transmission of payload divided into four 1 MB-frames should be more reliable than transmission of a single 4 MB-frame. From a statistical point of view, probabilities of these two events are equal. Thus, the fragmentation does not work if the retransmission process is not taken into consideration. This is explained by the following equation (2.1):

$$(1 - BER)^{4096*8} = (1 - BER)^{4*1024*8} \quad (2.1).$$

In general case, the equation can be represented in the following form (2.2):

$$(1 - BER)^{y} = (1 - BER)^{kx} \quad (2.2),$$

where: y – length of the long frame
x – length of the short frame
k – number of short frames required to carry the same payload like long frame.

The equation (2.2) is satisfied if $k = y/x$, and k, y, x are $\mathbf{N}^+$ numbers. This shows that the probability of a successful reception of four 1 MB frames is equal to the probability of successful reception of a single 4 MB frame. However, if a retransmission process is taken into the account, then some processing gain can be achieved from the payload fragmentation [44] (Figure 9). If a single bit error occurs in the long frame, then the entire 4 MB of data has to be retransmitted. If frame fragmentation is used, then only the defected fragment of the payload has to be retransmitted (1 MB).

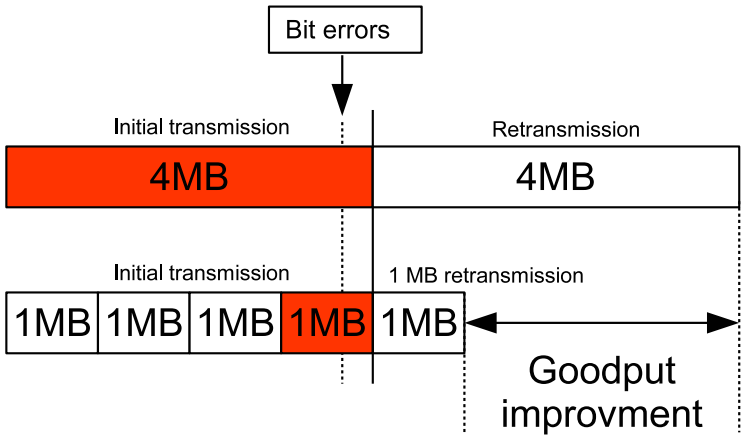


Figure 9: Explanation of improved transmission goodput achieved by fragmented frames. If a frame is fragmented and a bit error occurs in one of the fragments, then only the defected fragment has to be retransmitted but not the entire frame. Thus, the goodput is improved.

If the retransmission process is taken into consideration (ARQ), then the probability of successful transmission of a payload encapsulated into smaller frames is higher than probability of transmission of the same payload encapsulated into longer frames. The probability can be calculated by the following equation (2.3):

$$P(n) = \sum_{k=1}^{n} \left(\frac{n! * (1-(1-BER)^{l})^{n-k} * (1-BER)^{lk}}{(n-k)! * k!} \right) \quad (2.3),$$

where:

P(n) – probability of successful frame delivery after *n* transmissions,
l – frame length in bits,
BER – bit error rate.

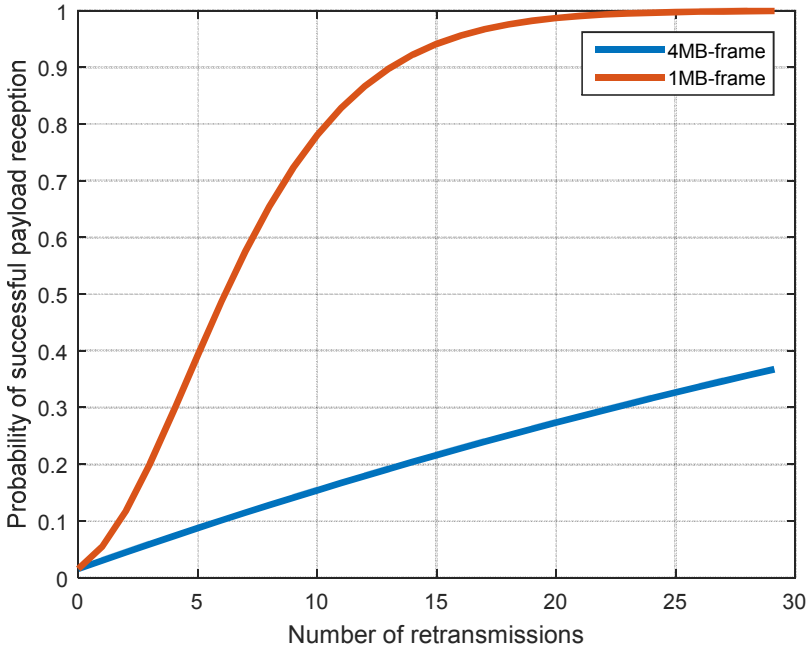


Figure 10: Probabilities of successful transmission of 4 MB- and 1 MB-frames as a function of the number of retransmissions at a constant BER = 1e-6. Shorter frame achieves higher probability of successful reception. If a frame is carrying less bits, then cumulative defection probability is lower.

Figure 10 compares the probability of a successful payload delivery of 1 MB and 4 MB frames as a function of the number of retransmissions. Shorter frame achieves significantly higher probability of successful reception. Such comparison does not include the transmission time. Transmission of the 4 MB-frame requires the period to be four times greater than transmission of the 1 MB-frame. Thus, improvement of the transmission performance of the 1 MB-frame is even higher than presented in Figure 10. This is shown in Figure 11, where the transmission time is taken into consideration instead of the number of retransmissions.

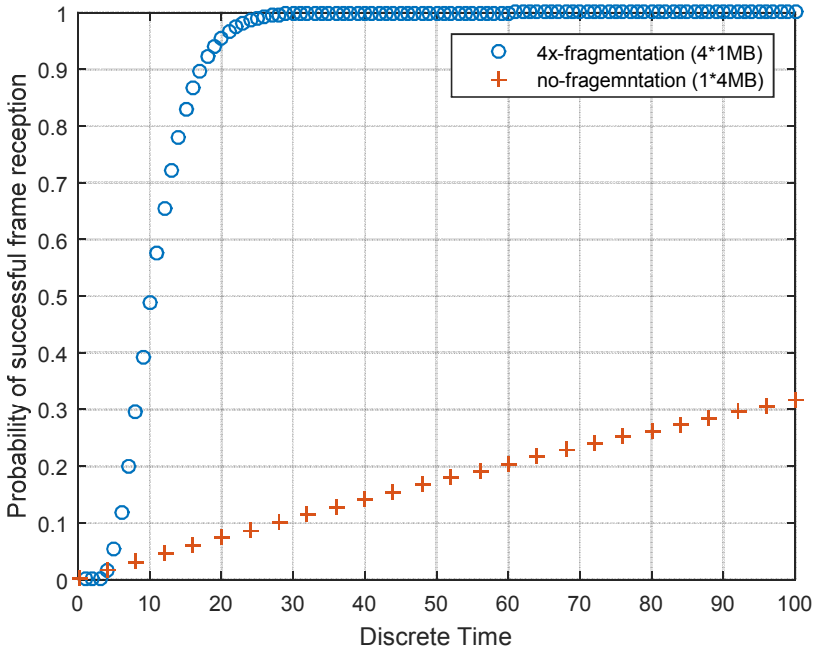


Figure 11: Probability of successful data transmission for 4MB- and 1MB-fragmented frames as a function of discrete time. Retransmission of smaller fragments is more effective than retransmission of the entire frame.

2.2 Frames aggregation and selective fragment retransmission

Frames fragmentation improves transmission goodput by decreasing frames length in a noisy environment. This reduces frame error rate, but there are also negative aspects of this process. Increased number of frames requires more preambles generated on the PHY level. Each frame is extended by a PHY preamble to find correct RF-gain (AGC – automatic gain control), synchronize the center frequency (AFC – automatic frequency control), and to recover the data clock on the receiver side. It means that the preamble has to be long enough to perform mentioned processes on the receiver side. In this time, user data is not transmitted and the preamble is reducing transmission goodput. Additionally, a frame header has to be added to each frame to signalize the frame length and data representation (e.g., used FEC code). Therefore, the number of transmitted preambles and headers has to be reduced. The only way to do it is to extend the frame length as much as possible. This reduces the number of transmitted preambles and headers, but very long frames are not preferred in a noisy RF-environment due to increased frame error rate. This causes an impasse, but there is a possibility to reduce the number of transmitted

preambles as well as reduce the logical frame size using frames aggregation and selective fragment retransmission [45] (Figure 12).

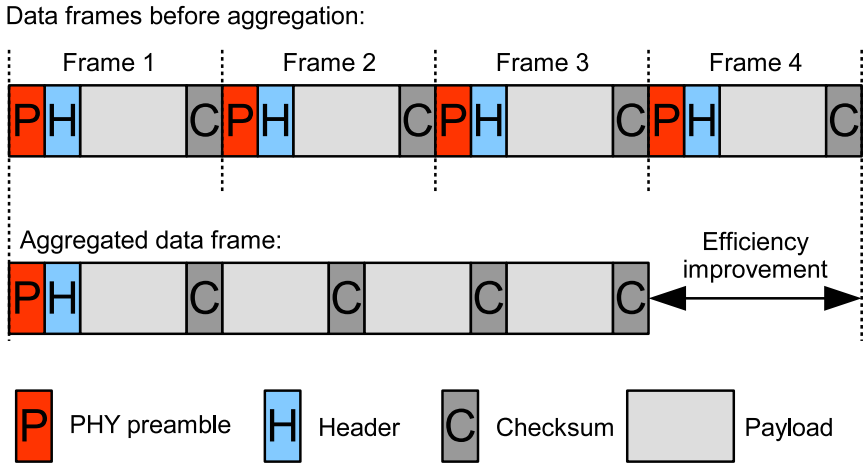


Figure 12: Frames aggregation. By using aggregation method, the number of transmitted PHY preambles and headers is reduced. Thus, protocol goodput (efficiency) is increased. Figure adapted by author from [45].

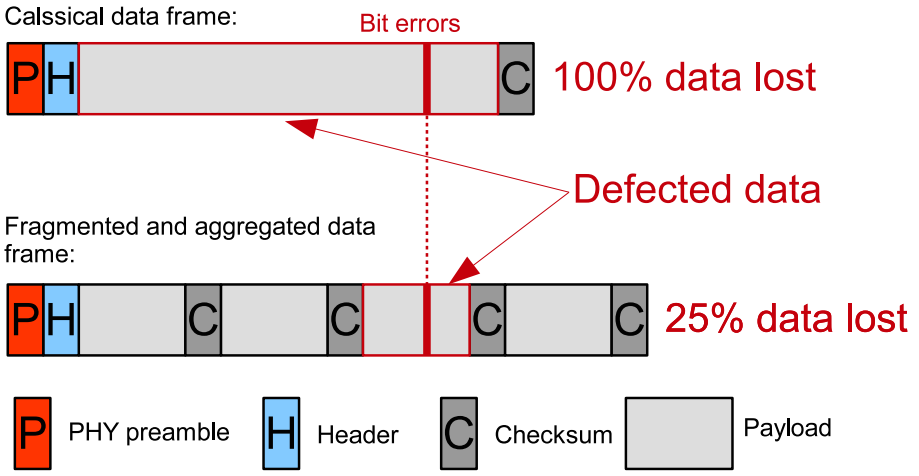


Figure 13: Frames aggregation and fragmentation. If both methods are applied, then the transmission goodput is improved due to limited number of transmitted preambles and headers. Additionally, in case of bit errors, only the defected frame-fragment is retransmitted instead of the entire frame (selective fragment retransmission).

To give a better overview of the problem, a system based on the maximal Ethernet MTU-size of 1500 octets and PHY parameters defined in the 802.11ad WLAN standard [7] is considered (preamble time - 1891 ns, data rate ~7 Gbps). Transmission of 1500 Bytes at 7 Gbps requires ~1714 ns. The preamble time increases this value up to ~3605 ns, so the average goodput is reduced to ~3.3 Gbps. Therefore, the 802.11ad standard uses frames aggregation to avoid such situations. The most important aspect of an aggregated frame is resistance to bit errors and reduced preamble-overhead. The fragments of the frame share a common preamble and header, but CRC fields are separate. The CRCs are recalculated for each fragment independently, which allows detection and retransmission of the defected parts individually (i.e., selective fragment retransmission [44]). There is no need to retransmit the entire frame as long as the frame header can be successfully decoded. Figure 13 demonstrates an approach based on the aggregation with fragment retransmission. In case of bit errors, the fragmented and aggregated frame achieves significantly higher transmission goodput. Additionally, fragments length can be controlled on the fly according to channel BER.

2.3 Automatic repeat request (ARQ)

Automatic repeat request (ARQ) [46] is one of the most important techniques used in wireless communications. The ARQ provides robustness in wireless protocols. Every time, when an incorrect frame is received, the ARQ uses a return channel to inform the transmitter about the lost frame. After that, the transmitter can schedule the frame for retransmission (Figure 14).

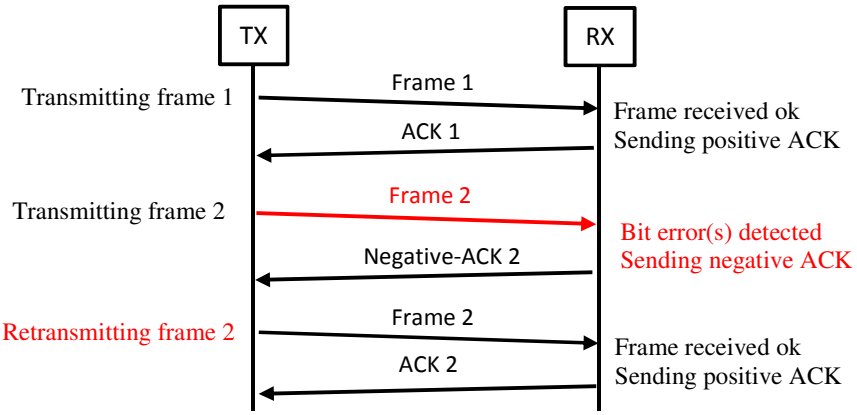


Figure 14: Stop-and-wait-ARQ. The receiver sends an individual ACK-frame after each received data-frame. The receiver switches from RX to TX mode, sends 1-bit-ACK message with an individual preamble, returns to RX mode, and waits for the next data-frame. Every ACK-preamble and RF-turnaround induces significant overhead to the transmission. Thus, transmission goodput is significantly reduced. Figure adapted by author from [46].

The stop-and-wait-ARQ solution (Figure 14) is inefficient. Both RF-frontends have to switch to transfer the acknowledge frame (ACK) after each data frame transmission. Additionally, the data frame has to be fully processed and the CRC has to be recalculated before the ACK-frame can be prepared[6] and sent. Data frame processing may introduce significant delay due to forward error correction processing (FEC) and pipelining. That reduces transmission goodput, which can be estimated by the following formula (2.4):

$$\eta = \frac{t_{data} * (1-BER)^{l}}{t_{overhead} + t_{data}} \qquad (2.4)$$

where:

l – frame length in bits
BER – bit error rate
t_{data} – time used for payload transmission
$t_{overhead}$ – time used for all other processing, e.g., radio switching; preamble, header and CRC transmission.

To achieve higher goodput, a different ARQ method has to be used, e.g., selective-repeat ARQ [46] (Figure 15).

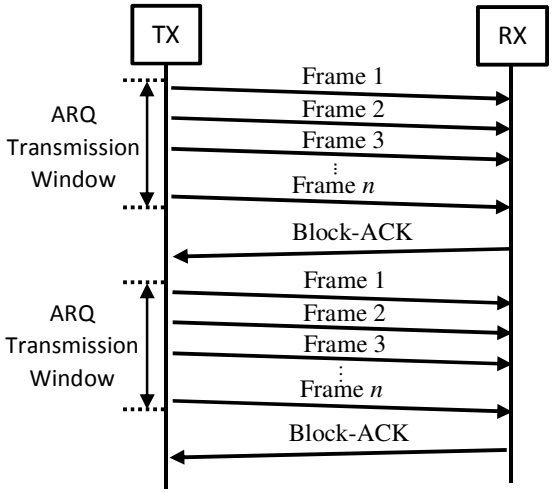


Figure 15: Selective-repeat ARQ. The ACK-frame is sent after 'n' data frames and all 'n' frames are acknowledged at the same time. This approach significantly reduces a number of transmitted ACK-frames (PHY-preambles) and RF-turnarounds. Figure adapted by author from [46].

[6] Preparation of the ACK-frame includes ACK-compression, FEC encoding, and CRC calculation. ACK compression schemes are described in section 3.9.

The selective-repeat ARQ repeats individual frames and uses a single block-ACK-frame[7] [47] to acknowledge all successfully received data frames. This reduces the number of PHY turnarounds and transmitted ACK-frames.

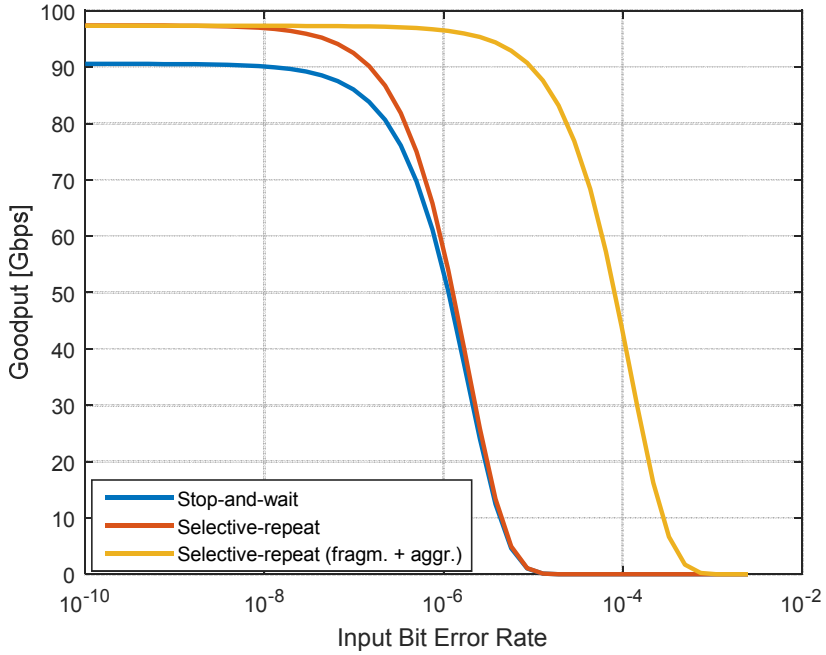


Figure 16: Comparison of ARQ methods. The selective-repeat method is used with 1 kB fragmentation and aggregation. The goodput on 'bad' links (~1e-5) is significantly improved due to selective fragment retransmission. 802.11ad PHY parameters are used to simulate the results [7].

Figure 16 compares achieved results of the ARQ methods. All RF-frontend parameters used for the estimation are taken from the 802.11ad standard [7] (preamble time – 1.891 us, RF-turnaround time – 1 us, data throughput – 7 Gbps). Additionally, the frame size is set to 64 kB, the number of frames in a single selective repeat ARQ transmission window to 64, and the frame-fragment size to 1 kB (according to chapter 3).

[7] The block-ACK contains information about the reception of the all transmitted frames through a corresponding bitmap, and it is transmitted after an explicit transmitter request. The construction of the bitmap is described in section 3.9.

2.4 Forward error correction (FEC)

Forward error correction (FEC) [46] is a method used to correct bit errors in received frames. The transmitter adds some redundant bits to transmitted frames, so that the receiver can use these bits to locate and correct bit errors. Selection of the optimal FEC code is difficult and channel dependent. Block codes and convolutional codes represent two main FEC categories. The block codes are processing fixed-size blocks of data. Each data block is individually extended by redundancy bits during the encoding process. There is no dependency and no overlapping between the blocks.
Block codes may have impact on the fragmentation scheme described in section 2.1 (Frames fragmentation). The fragment length can interact with FEC block length. The most instantly recognizable block codes are Hamming codes, Reed-Solomon (RS) codes, Bose-Chaudhuri-Hocquenghem (BCH), and low-density parity-check (LDPC) codes [46].
In contrast to the block codes, convolutional codes work on a continuous bit stream. The stream is processed inside a sliding window that continuously overlaps and moves by 1 bit. The redundancy information of the currently processed bit is spread over few neighbor bits. The number of the affected neighbor bits is defined by the sliding window length (constraint length [46]). The longer the constraint length, the larger the number of parity bits that are influenced by any given message bit. A larger constraint length generally implies a greater resistance to bit errors but requires more computation power and hardware for decoding. The termination of the stream requires a special method like tail-biting or bit-flushing [46].
The code rate of a FEC code defines the ratio between the original message length and the length of the message after encoding. The encoded message is usually denoted by a '*code word*'. Thus, code rate '*R*' of a FEC code is defined by '$R = k / n$', where the '*k*' is the length of a message, and the '*n*' is the length of a code word ('*R = length_of_a_message / length_of_a_code_word';* the *'R'* value is always < 1). For higher *R*-values, the decoder adds less redundancy bits to the message. Therefore, the FEC code achieves higher goodput but error correction performance is reduced.

2.4.1 Viterbi decodable convolutional codes

Viterbi decodable convolutional codes encoder consists of a few flip-flops and xor gates (Figure 17).
The encoder in Figure 17 generates a non-systematic code with code rate equal to $R = ½$. It means that the input sequence of length n is converted to a new sequence of length $2n$. The message bits are not part of the output sequence, and the data cannot be extracted from the sequence without decoding (the data stream on the input is replaced by the encoded output sequence). The decoding is much more complicated and can be performed by the Viterbi algorithm [48] invented by A. Viterbi in 1967.

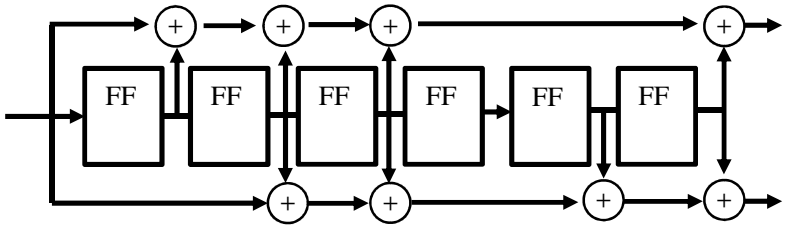


Figure 17: NASA convolutional encoder with polynomials (171,133). Each data bit shifted to the encoding circuits produces two bits of encoded stream. Thus, the code rate is equal to R= ½. The output sequence depends on 7 last bits, so the constraint length is equal to 7 (number of the delay elements plus 1). Figure adapted by author from [49].

Convolutional codes can produce an encoded stream with relatively low code rates, e.g., 1/2, 2/3. To increase the effective goodput of the code, puncturing patterns can be used. The puncturing process removes some bits from the stream in a predefined order. Thus, the code rate increases and it is possible to achieve many derivative codes. For example, an R = 8/9 code can be derived form a 2/3 base code. In such situation, every fourth bit is removed from the encoded stream. The convolutional codes prefer uniformly distributed single errors, and an interleaver is required for burst error correction.
The codes are widely used, therefore Viterbi decoder IP cores can be purchased from Xilinx [50], Altera [51], or even downloaded for free from the OpenCores website [52]. A standard decoder implementation achieves ~200 Mbps on a high-end FPGA [50]. Example of the applications, where the codes are used, are 802.11g WLAN (code rates: 1/2, 2/3, 3/4 [53]) and DVB-T (first generation) [54] standards. A detailed investigation of Viterbi decodable convolutional codes is out of the scope of this work. More information can be found in [46] and [49]. In [50], [55]–[57] detailed error correction characteristics against Eb/N0 are shown. Moreover, decoding computation complexity is investigated according to Viterbi decoder parameters (code rate, constraint length, traceback length, and decoding latency).

2.4.2 BCH codes

BCH codes [46] work differently from convolutional codes. Firstly, the BCH codes operate on blocks of length 2^n-1 and not on continuous streams. Secondly, the codes use polynomial operations over Galois fields (GF) to find and correct errors. Thirdly, an exact predefined number of bit errors can be corrected. Those errors, which either have burst or single characteristics, can be randomly distributed in a block. Figure 18 shows examples of arbitrary selected systematic BCH codes. Figure 19 shows the relation between error correction capability and the number of redundancy bits for three arbitrary selected BCH codes. For longer BCH blocks, more redundancy bits are required to correct the same number of bit errors.

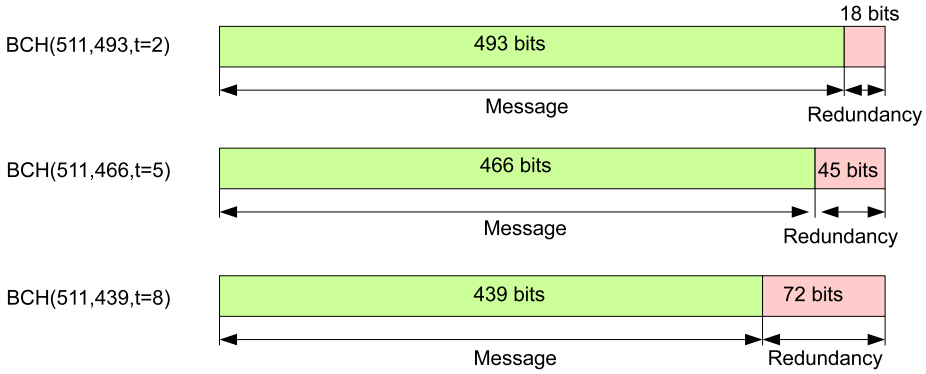


Figure 18: Examples of BCH-encoded code words. The encoder adds redundancy bits at the end of the message[8]*. The number of redundancy bits depends on the message block length and error correction capability (t). For 'small t-values'* [58]*, the number of redundancy bits can be estimated by the following formula: '⌈log₂(message_length_in_bits)⌉ × t'* [58].

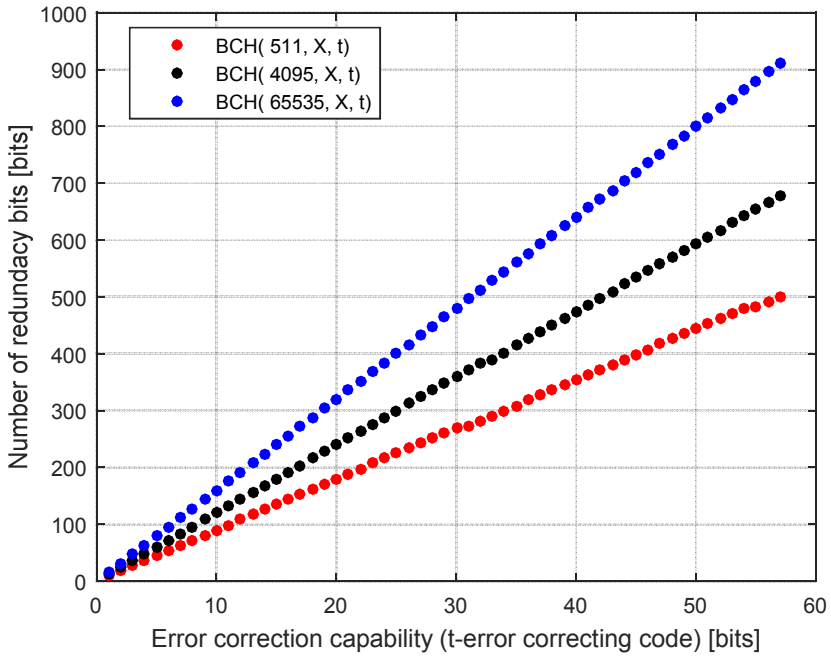


Figure 19: Relation between error correction capability and the number of redundancy bits for three arbitrary selected BCH block lengths: 511, 4095, and 65535 bits. For longer BCH blocks, more redundancy bits are required to correct the same number of bit errors.

[8] Investigation presented in this work is limited to systematic BCH codes only.

Some of the applications, where BCH coding is used, are new broadcast television standards: DVB-T2 [59] and DVB-S2 [60]. In both cases, a concatenated inner LDPC code with an outer BCH(65535,65343, t=12) code are applied.

2.4.3 Reed-Solomon codes

Reed-Solomon (RS) codes [46] are a subset of non-binary BCH codes class and have similar attributes to the BCH codes [61], [62]. RS codes correct random errors, and the error correction capability can be precisely defined like in case of BCH codes. However, the best performance is achieved against burst errors. The codes operate on symbols formatted into blocks. A typical size of a symbol is 8-bit, but it is possible to construct codes with shorter and longer symbols. Figure 20 shows examples of arbitrary selected systematic RS codes.

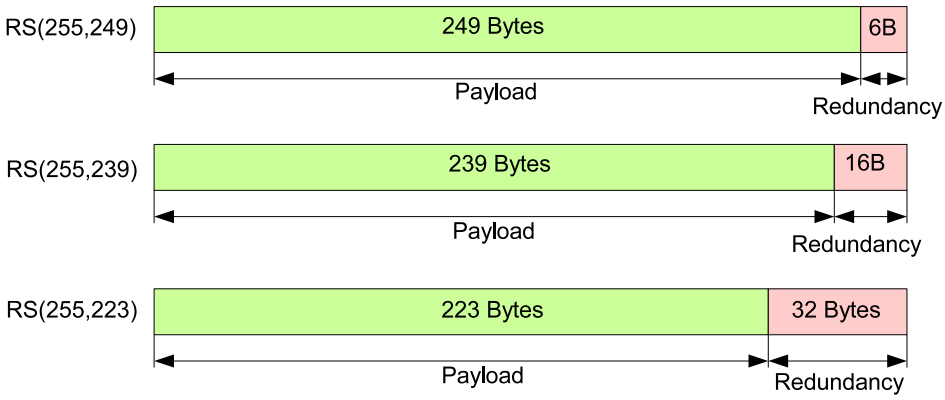


Figure 20: Examples of systematic RS code words. The RS codes correct up to 't' symbols, and require '2t' redundancy symbols. In the figure, GF(2^8) Reed-Solomon codes based on 8-bit symbols are shown. Symbol correction capability (t) is equal to 3, 8, and 16 bytes respectively for the presented codes.

In Figure 20 three RS code words are shown: RS(255,249), RS(255,239), and RS(255,223). The numbers define the RS code word size (n = 255 symbols) and the payload size (k = 249, 239 or 223 symbols). It means that the redundant information is defined as 6, 16, or 32 symbols (in this case 1 symbol = 1 byte). The symbol correction capability is defined by $t = \lfloor (n-k) / 2 \rfloor$. Thus, RS(255, 249) can correct up to 3 symbols (Bytes), RS(255,239) up to 8 symbols, and RS(255, 223) up to 16 symbols in the code word. The most important feature of RS codes is burst-error correction capability. The codes correct a whole symbol at the same time, and the number of erroneous bits in the defected symbol is irrelevant. It means that one symbol error occurs when just one bit in the symbol is defected or when all bits in the symbol are defected. Up to eight bits in the 8-bit symbol are corrected at the same time, and the cost of the correction in terms of redundancy symbols is the same. Two

redundancy symbols are required to locate and correct a single erroneous symbol. Figure 21 compares the required number of redundancy bits between BCH and RS codes to correct the same number of bit errors. RS codes require more redundancy bits to correct 1-bit single errors than BCH codes, but in case of 8-bit burst errors, RS correction performance is much higher than BCH codes performance.

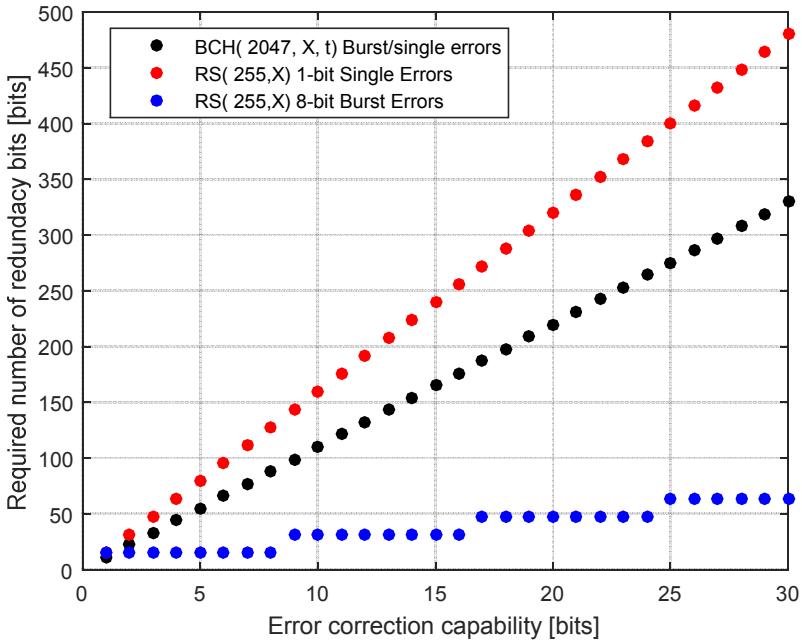


Figure 21: Comparison between BCH and RS error correction capability for single and burst bit errors. RS coding is not optimal for single errors correction (red markers) due to the symbol-oriented decoding. However, if RS codes are used against burst errors (blue markers), then RS decoder requires much less redundancy than BCH decoder (black markers).

One of the applications, where RS codes are used, is DVB-T terrestrial television system [54]. In the DVB-T standard, RS(204,188) coder is employed as an outer error correcting code for convolutional codes. Moreover, the codes are used in the new IEEE 802.3bj-2014 and 802.3by standards (25 and 100 Gbps Ethernet). The IEEE 802.3bj uses RS(528, 514) code calculated in GF(2^{10}) [63].

2.4.4 Reed-Solomon encoding algorithm

Coding and decoding procedures of RS and BCH codes are similar. Both codes use polynomial operations defined over GF fields. This sunbsection introduces the RS encoding algorithm. The operation can be represented as a polynomial division (2.5) [62]:

$$Redundancy\_poly = x^{n-k} \times Message\_poly \bmod RS\_generator\_poly \quad (2.5)$$

The redundancy bits are obtained from the remainder polynomial (*Redundancy_poly*). The required steps to calculate the redundancy bits are as follows. Firstly, the data has to be represented as a message polynomial. In this case, the *Message_poly* is defined by (2.6) [62]:

$$Message\_poly = M_{k-1} X^{k-1} + M_{k-2} X^{k-2} + \ldots + M_1 X + M_0 \quad (2.6).$$

The $M_{k-1} \ldots M_0$ are the message symbols [62] belonging to the GF(2^m), where the m is the RS symbol size. The message bit values have to be converted to the message symbols in the GF(2^m). This can be achieved using vector representation of the symbols [62]. In such case, data bits interpreted as vector representation define the message symbols.
The *RS_generator_poly* is a RS self-reciprocating generator polynomial of n-k degree. Generation of the polynomial is explained in [62].
The x^{n-k} is a displacement shift that converts the message polynomial of k degree to n degree polynomial, so the division is possible. As a result, a polynomial of maximum degree n-k is achieved. The coefficients of the *Redundancy_poly* polynomial define redundancy symbols. The redundancy symbols are converted to redundancy bits in the same way like the message bits are converted to the message symbols – the vector representation of the symbols is used [61], [62]. Figure 22 shows a schematic of an RS encoder.

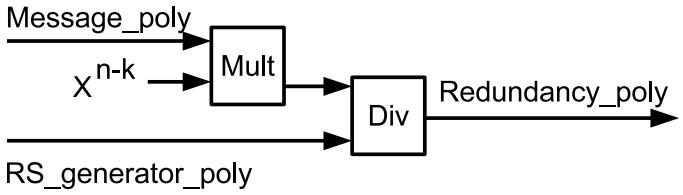


Figure 22: RS encoder schematic. Polynomial multiplication and division in GF arithmetic are required to calculate the redundancy bits. Figure adapted by author from [62].

Polynomial division can be implemented in hardware using a shift register circuit with GF additions and multiplications (Figure 23).

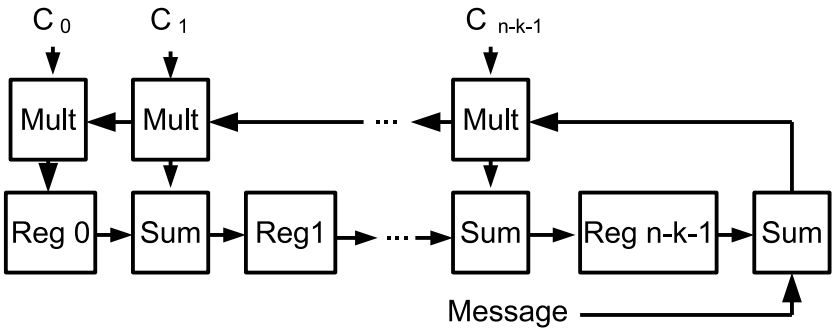


Figure 23: Shift register circuit for an RS encoder. The hardware implementation of an RS encoder requires a shift register with GF-additions and GF-multiplications. Figure adapted by author from [62].

In Figure 23, the C_0 ... C_{n-k} multiplication coefficients correspond to the coefficients of the RS generator algorithm. The redundancy symbols can be read from the registers *Reg 0* ... *Reg n-k-1* after shifting the message to the circuit.
The RS is a common coding algorithm, and therefore Altera and Xilinx FPGA vendors provide IP-cores for theirs products [64]. The RS(255,239) encoder achieves clock frequency up to 478 MHz on the state of the art Virtex7 FPGA and this corresponds to ~3.5 Gbps goodput.

2.4.5 Syndrome based RS decoding algorithm

Reed-Solomon decoding procedure is much more complicated than the encoding. The typical syndrome based decoding [65] is a five-stage process [62]:

1. Calculate 2*t* syndromes from the received code word.
2. Calculate error-locators (includes Berlekamps's and Massey's algorithm).
3. Calculate error locations (includes Chien Search algorithm).
4. Calculate error values.
5. Fix the received code word.

Step 2 is the most computation intensive operation in the decoding process [62], [66]. Figure 24 shows dependency between the decoding steps.

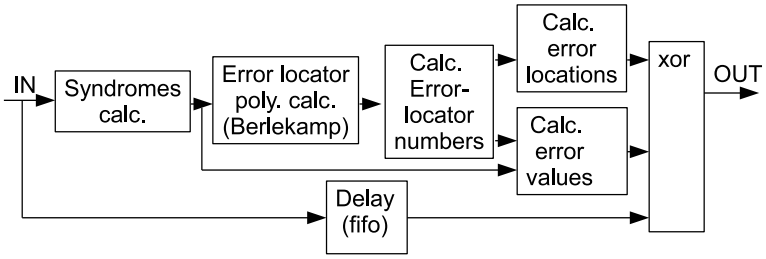


Figure 24: Schematic diagram of a syndrome-based RS decoder. Figure adapted by author from [61], [62].

In some publications (e.g., [66]–[70]), RS decoding process is represented as a three-step process: syndromes computations, key equation solving, and Chien search with errors evaluation. There exist several improvements of the state of the art decoder (e.g., decoding without using syndromes [67], improvements of the syndrome decoding circuit [65], improvements of the Berlekamp's decoding circuit [71]).

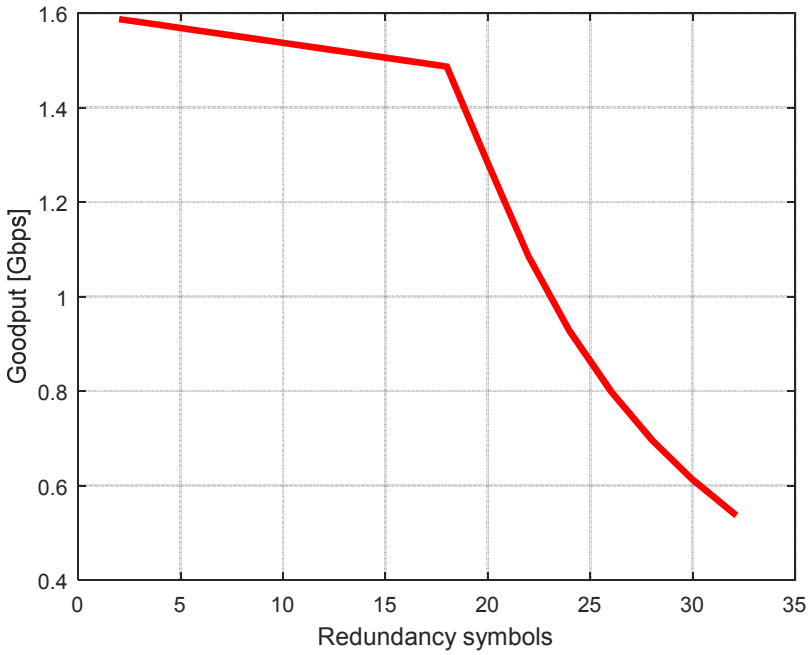


Figure 25: Decoding goodput of the Xilinx RS decoder IP-core. The goodput is significantly reduced for code words with 20 or more redundancy symbols ($GF2^8$).

Due to the popularity of RS algorithms, all leading FPGA vendors support IP-cores of RS decoders. A single instance of Xilinx RS(255,239) run on a Virtex7 FPGA achieves net data rate up to 2.2 Gbps. The decoder complexity and decoding latency is strongly correlated with the number of redundancy symbols. Figure 25 shows goodput of the Xilinx-RS decoder at 200 MHz as a function of redundancy symbols amount. Figure 26 shows the relation between processing latency and redundancy symbols [50], [64].

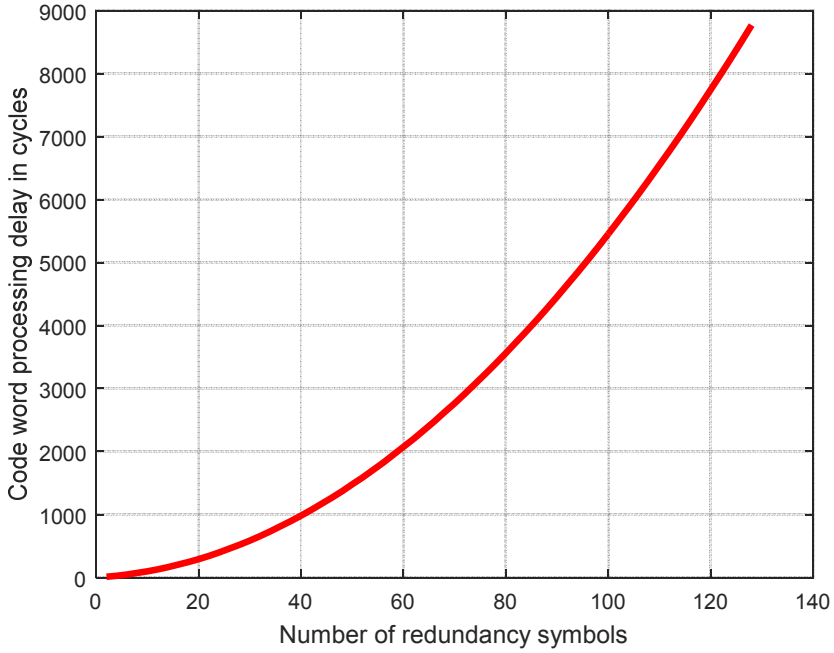


Figure 26: Processing latency of the Xilinx RS decoder as a function of redundancy symbols amount. Figure from data retrieved from [64].

2.4.6 Interleaved Reed-Solomon codes (IRS)

Interleaved Reed-Solomon (IRS) codes [72] use several RS coders aggregated in parallel (Figure 27). Such architecture has two advantages. Firstly, robustness against long-burst errors is improved (Figure 28). Secondly, throughput of the coder is multiplied. Thus, IRS decoder can be parallelized and scaled for 100 Gbps operations.

One of the applications, where interleaved RS codes are used, are compact discs (cross interleaved Reed-Solomon codes - 'CIRC' [73]). The ability of long error correction is used to compensate the effect of scratches on the CD surface. For that, two shortened RS codes are employed: RS(32,28) and RS(28,24) calculated in GF(2^8) [73].

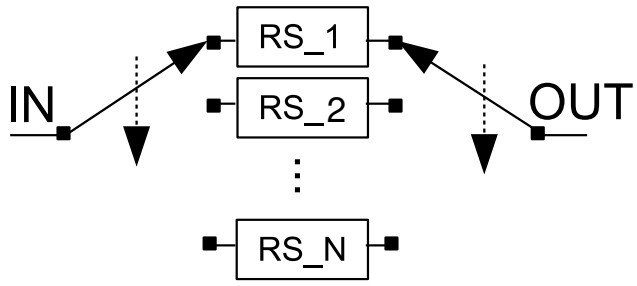


Figure 27: General structure of interleaved Reed-Solomon codes [72]. *The scheme uses 'N' RS coders to perform calculations. Such architecture improves processing throughput by 'N'-times and improves error correction performance against burst errors.*

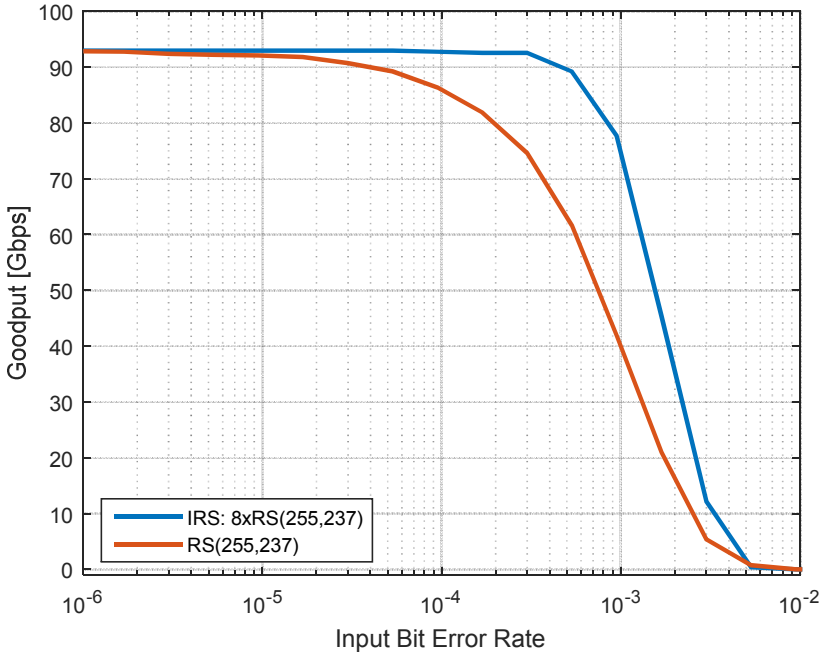


Figure 28: Comparison of burst-error correction performance obtained by a single RS decoder and an array of Interleaved-RS decoders. In case of IRS decoding, long sequences of bit errors are interleaved among multiple decoders, and therefore the effective number of erroneous symbols per decoder is reduced.

2.4.7 LDPC codes

LDPC codes are defined by a sparse parity matrix H [74], [75] (Figure 29a). The size and construction of the matrix directly influences the error correction performance

of the code based on the matrix. Thus, intensive research is addressed to find optimal algorithms to generate the matrix [76]–[78].
To explain the decoding algorithm, the LDPC parity matrix can be represented as a Tanner graph [75] (Figure 29b).

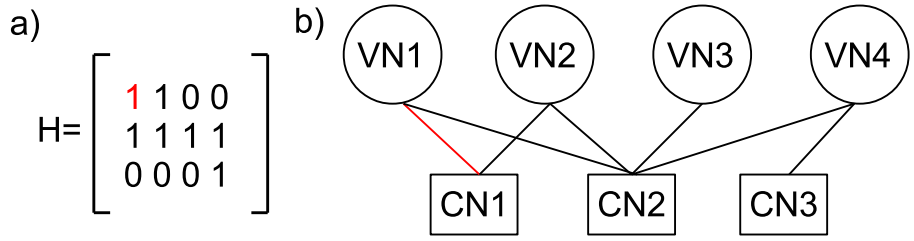


Figure 29: LDPC parity matrix (a) and a corresponding Tanner graph (b). The matrix element marked in red corresponds to the red path in the Tanner graph. Figure adapted by author from [75].

Each row of the matrix corresponds to a check node (CN) of the Tanner graph, and each column corresponds to a variable node (VN) of the graph. If $H(m,n) = 1$, then the variable node n (VN_n) is connected to the check node m (CN_m). For example, the matrix element (1,1) marked in red (Figure 29a) corresponds to the red connection in the Tanner graph (Figure 29b). LDPC decoding is based on messages passed between the variable and check nodes (known as 'belief propagation'). Firstly, variable nodes are initialized with received code word bit values. After that, the decoding algorithm starts. In every iteration, variable nodes send their values to check nodes (Figure 30).

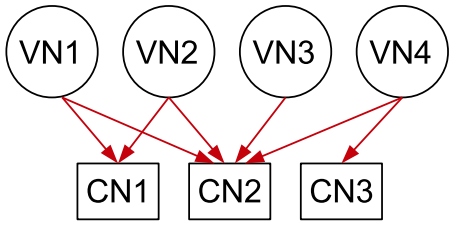


Figure 30: LDPC decoding (1) – passing estimated bit values to check nodes.

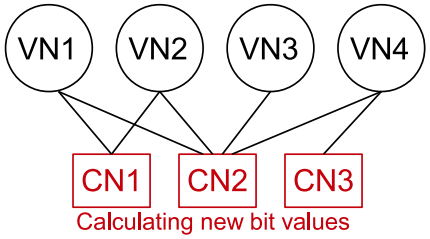


Figure 31: LDPC decoding (2) – estimating new bit values by check nodes.
Now, each check node processes the messages with a predefined formula (Figure 31), and sends back the calculated values to each of the connected variable nodes

(Figure 32). The variable nodes use the received values and combine them with their own values stored in local memories (Figure 33). In the next iteration, new values are sent to the check nodes and the process is iteratively repeated.

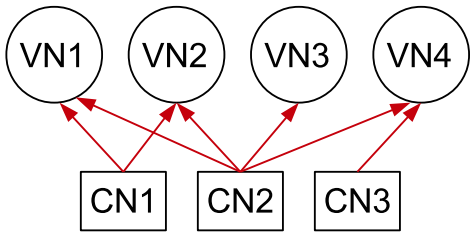


Figure 32: LDPC decoding (3) – sending newly estimated bit values to the variable nodes.

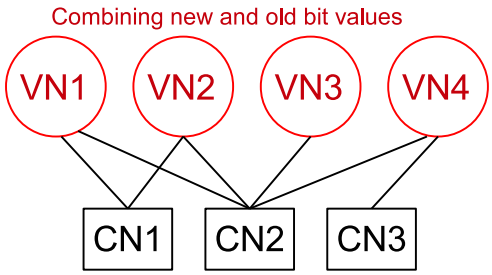


Figure 33: LDPC decoding (4) – combining new and old bit values in the variable nodes.

A very important precondition is that the message sent by the variable node does not depend on the message from the same variable node. Thus, only "extrinsic" information is used to calculate check node values in each step [74] (Figure 34).

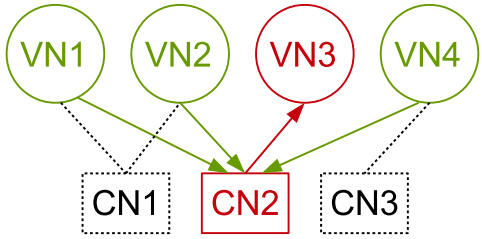


Figure 34: Only "extrinsic" information is used to calculate check node values in each step of LDPC decoding process. In the presented case, new value passed to VN3 (marked in red) is calculated by CN2 using information provided from VN1, VN2, and VN4 (marked in green) but not from VN3 (marked in red).

More details on LDPC coding can be found in [74], [75], [78]. Especially, check node's processing algorithms and the procedure of combining variable node values

are important. The procedures define error correction performance and calculation complexity of LDPC decoders. One of the key design aspects of the LDPC decoders is a tradeoff between calculation complexity, size of the parity matrix, and hardware resources.
LDPC codes are used in many modern communication systems (e.g., DVB-S2 [60], DVB-T2 [59], 802.11n [53], 802.11ad [79], WiMAX [80], 3GPP LTE [81]). In some applications, relatively big parity matrixes are used. For example, DVB-S2 in one of the operating modes uses a matrix size of 48600×64800 elements. Such matrix corresponds to a LDPC(64800,16200) code with a code rate of R = ¼ [60]. The decoder uses 64800 variable nodes and 48600 check nodes. This corresponds to 16200 data bits and 48600 parity bits per single code word.
One decoder implementation proposed for 802.11ad WLAN achieves up to 160 Gbps and is one of the fastest LDPC decoders in the world [75]. The solution is realized in 65 nm SVT technology, uses code word length of 672 bits (546 data bits + 126 parity bits), 9 hardware unrolled-iterations, min-sum algorithm [82], and accepts data symbols with representation quantized to 4-bits (soft decision decoding).

2.4.8 Interleaving

Some correction algorithms (e.g., Viterbi decodable convolutional codes) cannot be used for burst errors correction due to very poor correction performance for such a type of errors. Other algorithms, (e.g., LDPC) can be used for burst error correction, but correcting performance is reduced in such case. Thus, interleavers are used to divide burst errors to single errors before FEC decoding (Figure 35). The complete process is as follows: the data is interleaved before transmission, the interleaver mixes the data bits in a pseudorandom fashion. Such mixed bits are sent via communication channel, and burst errors are introduced in the pseudo-randomly mixed data bits (Figure 36). After reception, the receiver performs deinterleaving. All consecutive burst errors are converted to single errors, because mixing the data bits is performed in reverse order. After this operation, FEC decoder processes single errors instead of burst errors. Therefore, error correction performance is improved.

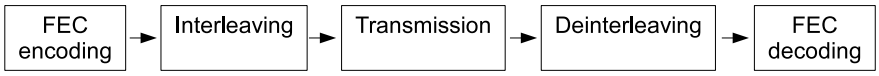


Figure 35: Example architecture of a system with FEC and interleaving.

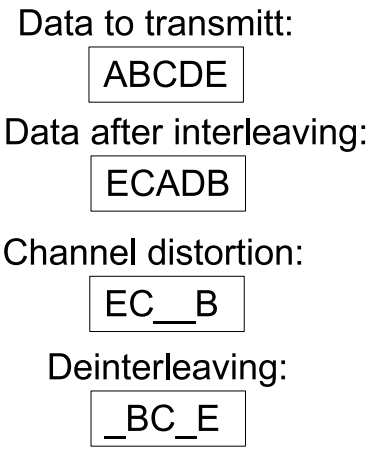


Figure 36: Example of interleaving and deinterleaving processes.

Two commonly used interleaver types are shown in Figure 37 and Figure 38 (convolutional and matrix interleaver respectively). The size of the interleavers (the number of memory elements) defines the permutation property of the structures. If more memory elements are used, burst errors are split over longer sequences. The interleaving and deinterleaving processes increase decoding latency of FEC encoding and decoding. Thus, the size of interleavers has to be selected carefully according to the length of the expected burst errors. For RS codes, symbol interleaving instead of bit interleaving has to be used. In the other case, error correction performance of RS codes for burst errors is significantly reduced and such a decoding has no sense.

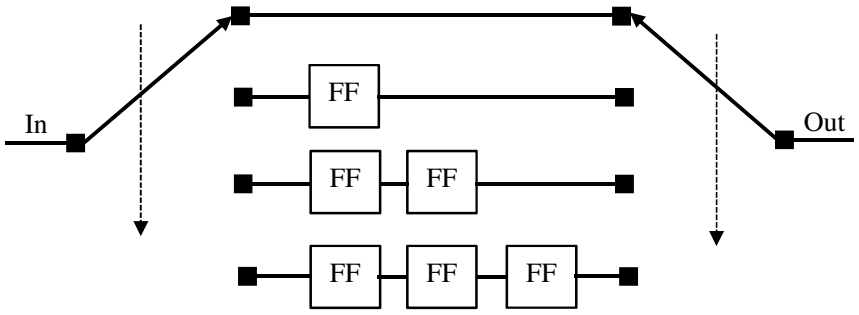


Figure 37: State of the art convolutional interleaver used to split burst errors to single errors. Figure adapted by author from [83].

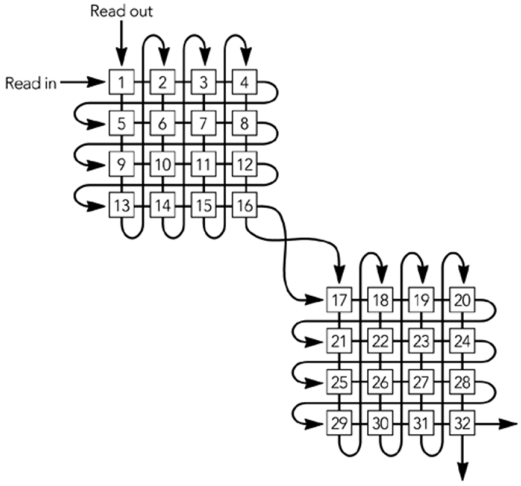


Figure 38: State of the art matrix interleaver used to split burst errors to single errors. Figure retrieved from [84].

2.4.9 Comparison of FEC and fragmentation

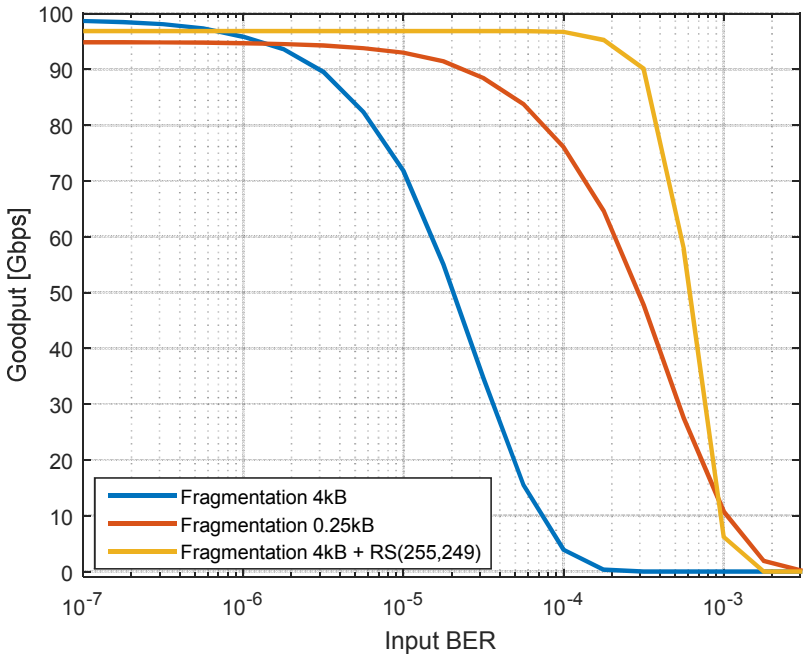


Figure 39: Performance comparison between frame fragmentation and forward error correction. In the considered case, RS(255,249) coding obtains higher goodput than fragmentation (0.25 kB).

Fragmentation and FEC techniques improve communication's goodput over noisy channels. The gain obtained by FEC is usually higher than the gain obtained by fragmentation, but FEC is more complex and requires more computation power than fragmentation. Results of both techniques are compared in Figure 39. Even 'simple' RS(255,249) coding allows stream processing with around 15 times higher BER than the 8× fragmentation denoted in red in Figure 39. It means, FEC is the key feature in case of improving communication reliability and is more effective than fragmentation. In the proposed 100 Gbps design, both features are used simultaneously to improve the operational BER range of the DLL processor.

2.4.10 Comparison of selected FEC codes

One important practical aspect of FEC algorithms is processing latency of the decoders. If a short data block is considered, then results of Hamming decoding can be ready in a single clock cycle. Decoding results of typical Viterbi and RS decoders are ready in a few tens or even in a few hundreds of clock cycles [50], [55], [64], [85]. RS codes that operate on long blocks introduce relatively long delays to the decoding pipeline. Very long decoding latency may introduce some difficulties for practical implementations. It may happen that transmission of a frame is finished before information extraction from the RS-encoded header. Usually, the frame header contains information necessary to decode the data (e.g., the length of the frame, the length of the data fragment, FEC and encryption schemes used for the data, etc.). If the mentioned frame attributes are unknown, the frame cannot be processed on the fly, but has to be buffered until the header is successfully decoded. This introduces idle cycles into the processing pipeline and requires ultra-fast cache memory with ultra-short access time[9]. The header decoding is especially difficult in 100 Gbps networks, where transmission of a frame header can be shorter than 2 ns. This corresponds to two clock cycles for an ASIC running at 1 GHz clock, but as mentioned before, decoding of FEC algorithms may take up to hundreds of cycles. This problem is deeply investigated in section 3.17, where accurate decoding timings for typical FEC algorithms are introduced.
Another important aspect is soft or hard bit representation of the demodulated signal. The hard-decision decodable (HD) codes process bits information quantized to '0' or '1'. The soft representation of a data bit is carrying additional decoding probability (Figure 40).

[9] The required characteristic of the cache memory is discussed in subsection 3.2.4.

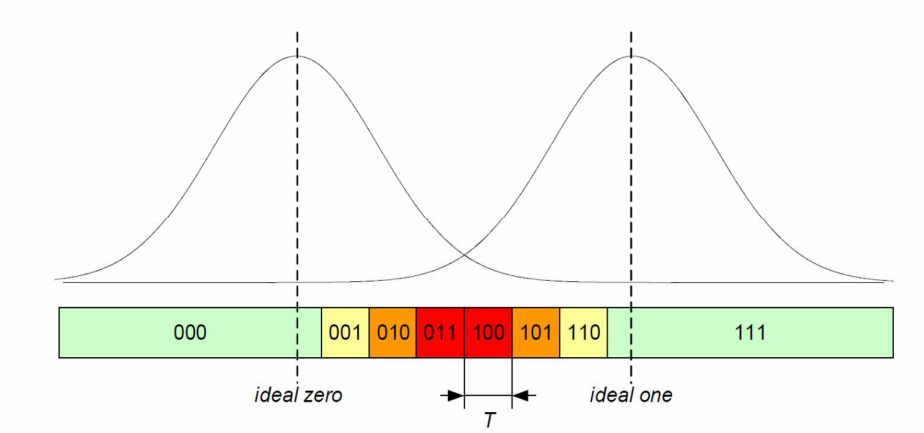


Figure 40: Quantization zones for a 3-bit soft decision demodulator. The demodulation confidence is represented by 3-bit soft-values. SD-FEC decoders include the information in the FEC decoding process to increase error correction performance. Figure retrieved from [49].

The soft information is used to increase error correction performance of FEC algorithms. The algorithms can use this information to locate all 'weak'-bits. If the analysis of redundancy information indicates some positions where bits are incorrect, then the unreliable bits are changed firstly during the decoding process. This significantly improves error correction performance, but on the other hand, soft bit representation requires more complicated decoding routines and very fast interconnection to an ADC for the soft-bit values. In case of soft encoding proposed in Figure 40, a 100 Gbps transceiver requires a throughput of the ADC of 300 Gbps. Such throughput is difficult to achieve and costly in terms of energy consumption.
For some of the FEC codes, hard- and soft-decision decoding routines are known. RS and BCH decoders usually use hard-bit representation (soft RS and BCH decoders are published in [86] and [87]). However, Viterbi and LDPC use soft-bits in most practical applications. Results of the hard- and soft-decision Viterbi and LDPC decoders are compared in Figure 41 and Figure 42. Hard decision (HD) method reduces error correction performance by ~2 dB for Viterbi and ~1.7 dB for LDPC (AWGN channel[10] is considered).

[10] Additive white Gaussian noise (AWGN) is a basic noise model used in Information theory to mimic the effect of many random processes that occur in nature.

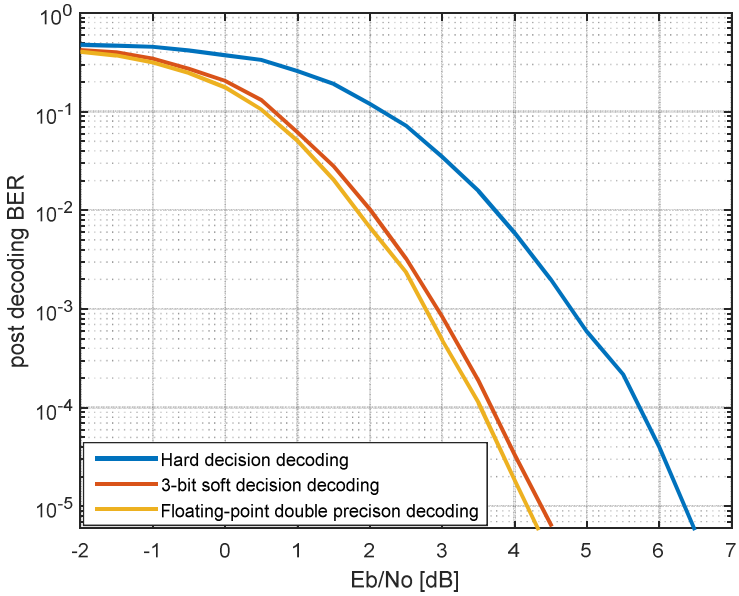


Figure 41: Comparison of error correction performance for hard- and soft-decision Viterbi decoders. The soft information included in the decision process improves the performance by approx. 2 dB.

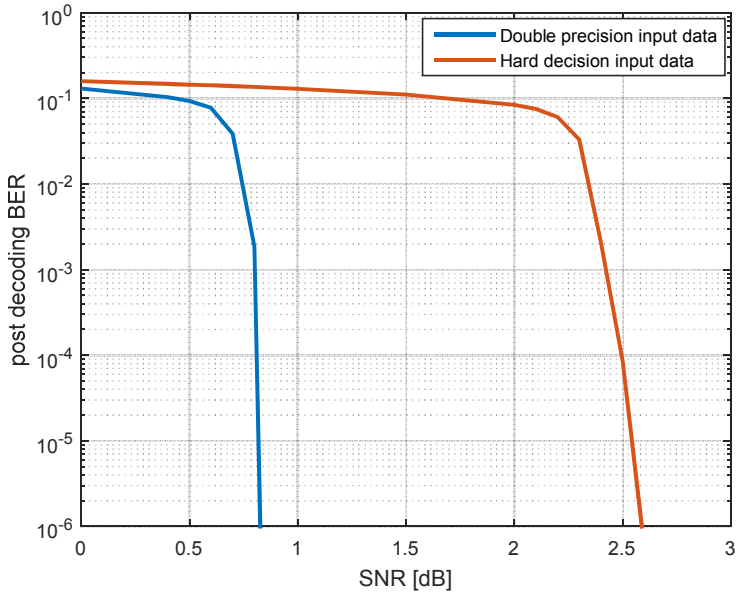


Figure 42: Error correction performance compared between soft- and hard-decoding LDPC. The soft information included in the decision process improves the performance by approx. 1.8 dB.

Figure 43 and Figure 44 compare error correction performance of LDPC, RS, BCH, and convolutional codes over an AWGN channel for two selected code rates (R = ½ and R = 8/9). R = ½ is often used for LDPC and convolutional codes, R ≈ 8/9 is commonly used for BCH and RS codes. All algorithms use hard-decision decoded information. For a code rate R = 1/2, LDPC achieves the best results. For a code rate R = 8/9 the situation is more complicated. The differences between the codes are less significant than in the first case. Convolutional coding achieves good results, but the slope of the representing curve is less sharp than for the other algorithms. Due to this reason, the LDPC decoder is again leading in the benchmark (according to section 4.6, BER < 1e-5 is considered). Therefore, LDPC codes are probably the optimal choice for the tested AWGN channel. If the channel model is changed and burst errors occur during the transmission, then the RS achieves significantly better results. RS algorithm corrects symbols, but not individual bits. Cost of a single bit and a single symbol correction is the same (two redundancy symbols are required for one corrected symbol). Due to this reason, RS codes are classified as multiple burst error correcting codes.

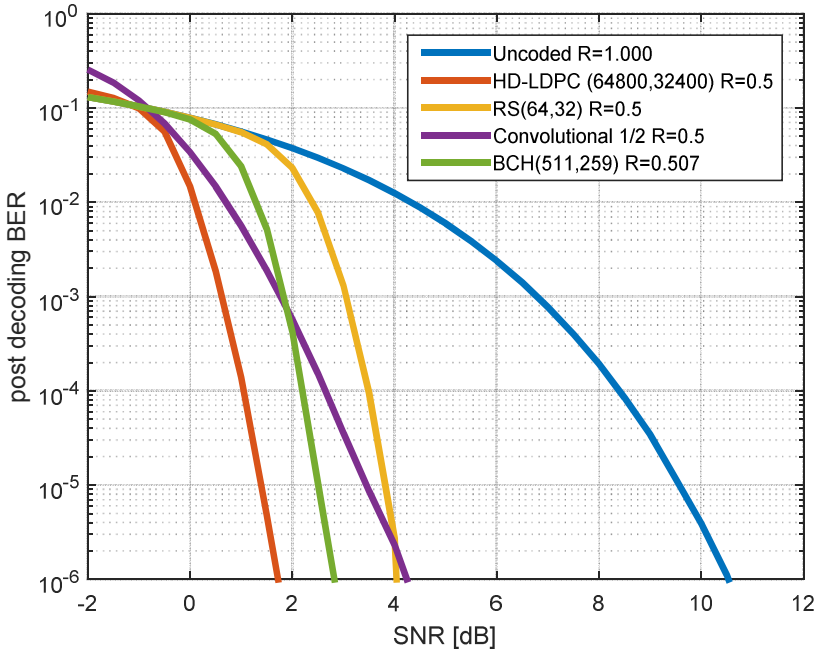


Figure 43: Comparison of error correction performance for basic FEC codes with code rate R ≈ ½. HD-LDPC decoder achieves the best results. The letter 'R' in the legend denotes code rate of the codes.

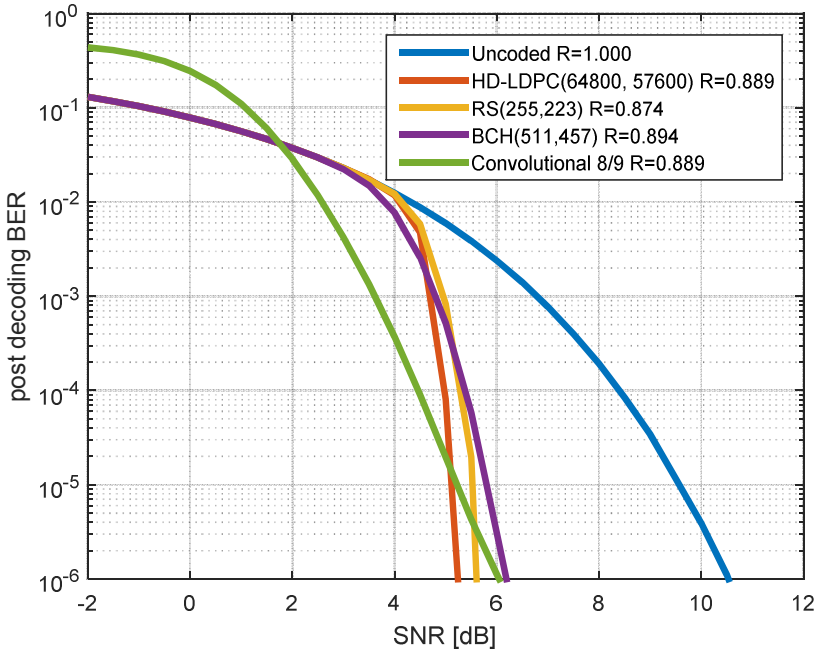


Figure 44: Comparison of error correction performance for basic FEC codes with code rate R ≈ 0.88. HD-LDPC decoder achieves the best results (BER < 1e-5 is considered). The letter 'R' in the legend denotes code rate of the codes.

Selection of the best code for the targeted 100 Gbps data link layer has to include much more aspects than discussed in this section. For example, the comparison has to include error correction performance for completely destroyed baseband symbols due to synchronization issues. In section 3.13, the same algorithms are compared in real implementation of the targeted DLL processor, including frames fragmentation, aggregation, selective fragment repetition, interleaving, and three different error characteristics. Moreover, decoding latency and decoding complexity is important as well (sections 3.17 and 3.18). Energy efficiency is one more factor to compare (sections 3.16.7, 4.4.2, and 4.4.3). Therefore, selection of the optimal coding is complex and is discussed in several sections of this work. Here, only a brief overview of the theoretical error correction performance of the selected codes is given.

2.4.11 Encoding and decoding throughput of selected codes

The decoding effort of FEC codes is probably the most demanding issue to solve during communication processor implementation along with FEC functionality. Figure 45 compares calculation speed of selected FEC codes on a general-purpose i7-4702MQ PC processor [88]. In the group of compared codes, RS(255,239) is the fastest algorithm and achieves ~3.8 Mbps. This simple example shows complexity of FEC decoding. FEC allows improving transmission goodput by reducing the

number of retransmissions, but calculation effort of FEC is challenging. Thus, in section 3.18, more efficient hardware implementations are proposed to deal with 100 Gbps FEC processing.

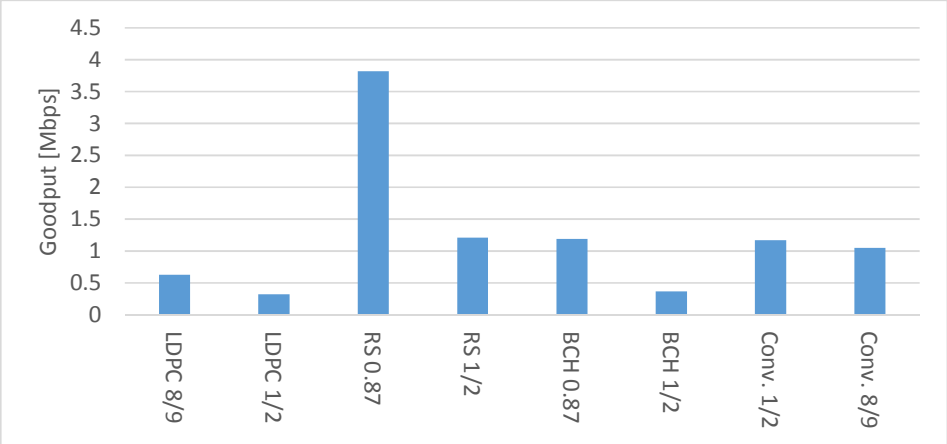


Figure 45: Comparison of coding goodput for selected FEC codes. The measurements are performed on a general-purpose i7-4702MQ PC processor (encoding and decoding, single thread, Matlab's Communications System Toolbox library).

2.4.12 Turbo product codes (TPC)

Some FEC schemes use two or even more error correction algorithms to improve error correction performance. Additionally, data can be interleaved between encoding iterations, such that message bits are mixed within different code words. Thus, information stored in code words mixes as well, and the effective code word length is increased due to dependencies between them (Figure 46). Moreover, decoding algorithm can be executed several times, and redundancy information is reused at different stages of the iterative decoding routine (Figure 47). Such approach utilizes redundancy information more effectively than a single pass-decoding scheme. Turbo product codes (TPC) [89], [90] use all of the mentioned processes all together. Figure 46 demonstrates a typical two-dimensional TPC code proposed for error correction in 100 Gbps optical networks [89], [90]. In this case, user data is written into a matrix of size equal to 357×357 bits. After that, the column encoder extends each column adding redundancy bits (usually a BCH code is used). The same encoder calculates redundancy for each row. After these two operations, the matrix size is extended to 390×390 bits. It means that every column and every row uses 33 redundancy bits to protect the data, so each bit of data is protected by two independent code words (the row code word and the column code word). After this operation, the matrix can be sent via a communication channel, and the receiver starts TPC decoding routine. The decoder calculates syndromes for each row and column. If a particular row or column contains errors, which can be corrected, then the decoder corrects the errors and the successfully decoded code word is written back

to the matrix. If the number of errors exceeds error correction capability of the used code, then the code word remains unchanged. After the first iteration, the decoding of rows and columns is repeated in the second iteration. Some errors were corrected in the first decoding pass. Thus, some defected code words, which were not decodable in the first iteration, can be successfully decoded in the second pass. The number of errors in the matrix is reduced with each iteration, until the matrix is successfully decoded, or a stall pattern [91] occurs. The stall pattern is a cluster of errors, which cannot be corrected by the row, nor by the column decoder. If e.g., a 3-error correcting BCH code for rows and columns is used, then 16 errors clustered in a rectangle shape will block the decoder (Figure 48).

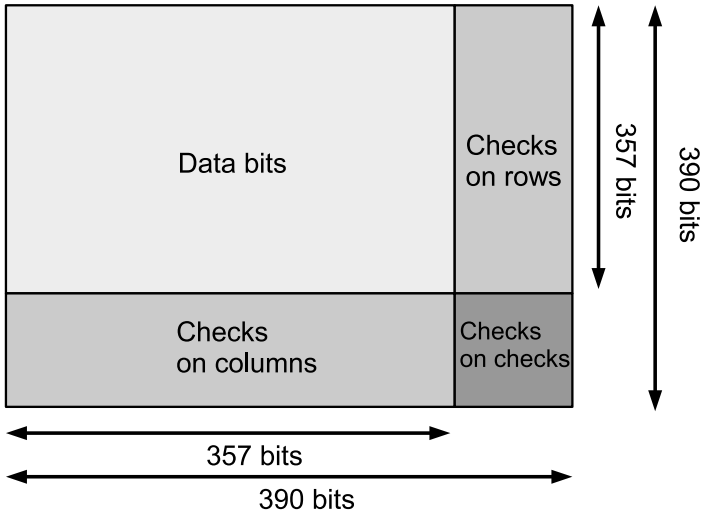


Figure 46: Turbo product code (TPC) decoding matrix. Vertical and horizontal code words protect data bits. Figure adapted by author from [90].

The iterative decoding of rows and columns can be alternatively presented as shown in Figure 47. Here a BCH code is assumed for row and column encoding.

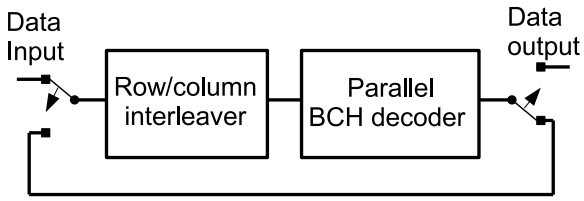


Figure 47: Alternative representation of a TPC decoder. The row/column interleaver transposes the serialized data matrix.

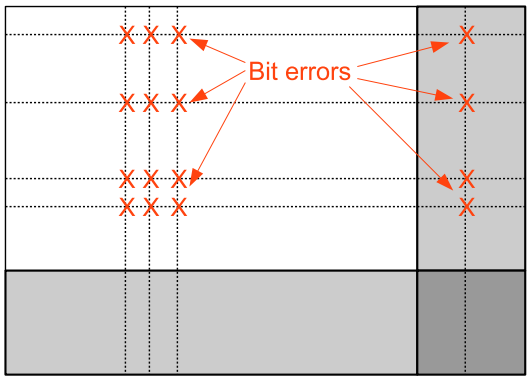


Figure 48: Stall pattern. If a 3-bit error correcting BCH code for rows and columns is considered, then 16 errors clustered in a 4-by-4 rectangle pattern blocks the decoding process. Figure adapted by author from [89].

2.5 Hybrid automatic repeat request (HARQ) and link adaptation

Hybrid automatic repeat request is a combination of FEC and ARQ[11] techniques [92]. Transmitted frames are encoded using FEC algorithms and sent along with redundancy data. The receiver firstly tries to correct the errors occurred during the transmission (FEC processing). If it is not possible, then retransmission is requested (ARQ processing). The FEC and ARQ can be mixed in different ways. The simplest version is the HARQ type I (denoted as HARQ-I), where static FEC coding is applied for all frames [93] (Figure 49).
HARQ-I method reduces transmission goodput on 'good' channels, where the FEC redundancy is unnecessary, or it is used only occasionally. An improved version named HARQ type II (denoted as HARQ-II) [94], is a more sophisticated algorithm. It sends the redundancy bits only if the receiver demands them (Figure 50). Initially, the frame is sent without any data for error correction. If the initial transmission was successful, the redundancy data will not be sent, which significantly improves goodput on 'good' links. If the initial transmission fails, the receiver requests redundancy bytes, and the transmitter sends the redundancy data in an additional frame. This shows an important difference between the type I and type II methods. In HARQ-II, two frames transmissions are necessary to correct the defected frame. In HARQ-I, the FEC is included in the data frame and no further transmissions are required. This improves goodput on 'bad' links, where FEC is necessary to successfully decode frames.

[11] ARQ processing is explained in section 2.3.

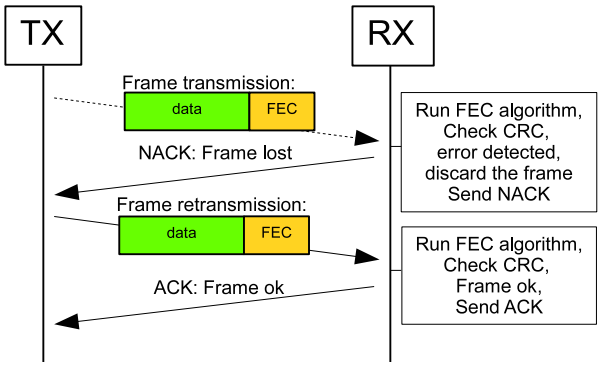


Figure 49: HARQ-I algorithm. FEC redundancy bits are added to each transmitted frame. If a frame is delivered without bit-errors, then the redundancy bits are not used in the decoding process. Thus, the bits induce overhead and reduce goodput.

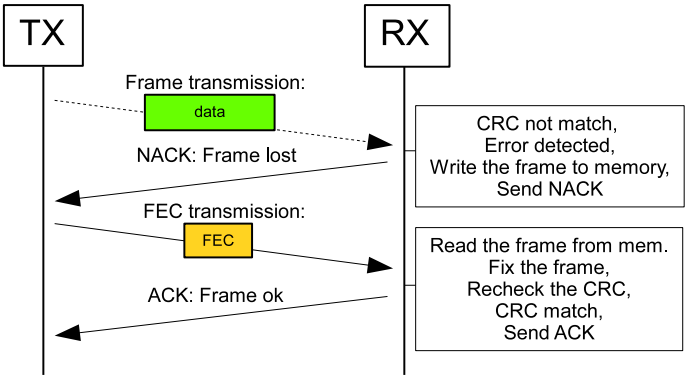


Figure 50: HARQ-II algorithm. The FEC redundancy bits are sent upon receiver request. Thus, if the receiver gets frames correctly, then the redundancy bits are never sent, and the transmission goodput increases.

HARQ type III [94] is an even more advanced version of the HARQ-II. In HARQ-III, FEC retransmissions are self-decodable, i.e., each retransmission can be decoded individually without any dependencies on previous transmissions. This can be achieved using a *1/n* convolutional code (R = 1/n) with n number of puncturing[12] patterns. For every retransmission, a different puncturing pattern is used, and a unique sequence of bits is sent (Figure 51). It is easy to demonstrate this technique with e.g., 1/3 convolutional code, where the following bits are sent in the ascending order of transmissions:

[12] Puncturing is the process of removing some of the encoded bits after FEC encoding.

- 1st transmission: bits on positions 1, 4, 7, 10, 13, 16, 19, (…) are transmitted, other bits are deleted; puncturing pattern = [1, 0, 0];
- 2nd transmission: bits on positions 2, 5, 8, 11, 14, 17, 20, (…) are transmitted, other bits are deleted; puncturing pattern = [0, 1, 0];
- 3rd transmission: bits on positions 3, 6, 9, 12, 15, 18, 21, (…) are transmitted, other bits are deleted; puncturing pattern = [0, 0, 1];

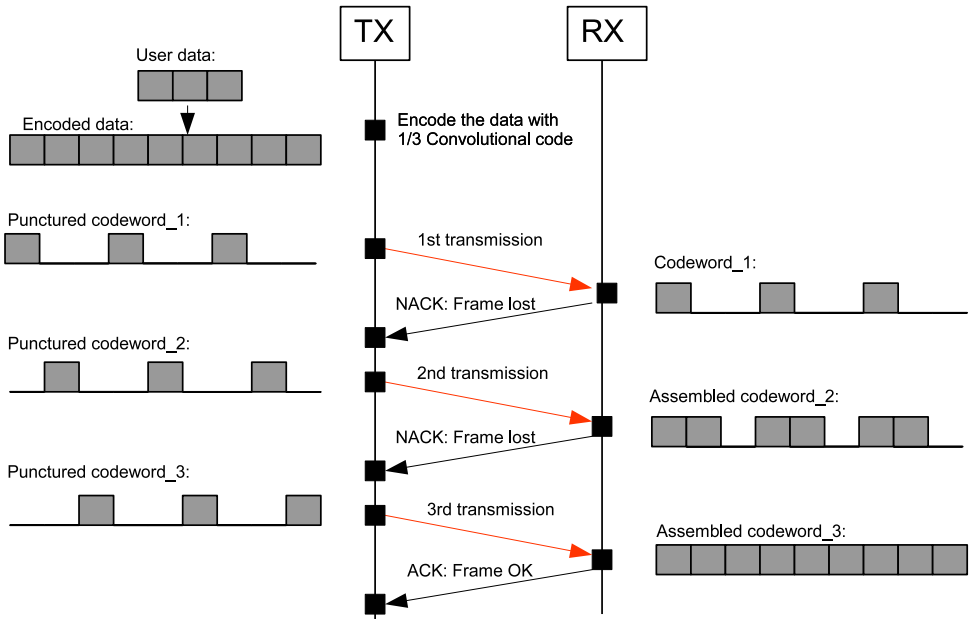


Figure 51: HARQ-III algorithm. Each retransmission is self-decodable and uses different set of redundancy bits. Such strategy allows the decoding of each retransmission individually. If this fails, then the receiver can combine information from previous retransmissions. After combining, more redundancy bits are available and the probability of successful code word decoding is higher.

There is no redundancy in the first transmission, thus the method is efficient in good links (code rate $R = 1$). If the initial decoding fails, then the second portion of bits is sent. Firstly, the decoder tries to decode newly received data. If the decoding is successful, then no further actions are required. If the decoding fails, then data from the first and second transmission is concatenated and decoded (code rate R = 1/2). If the frame still contains bit errors, then the third sequence is sent. After that, all bits are transferred and the complete not-punctured sequence can be decoded (code rate $R = 1/3$).
There is an additional option for the HARQ methods called chase combining. In the chase combining approach, the same set of bits is retransmitted, and soft values of the received bits are added together using a maximum ratio combining method (MRC). The MRC is a weighted average of the soft bit values, where a signal to

noise ratio (SNR) of the received signal is the proportional weight for averaging. Such combining increases the resulting SNR of bit values, by adding the signal from repeated-transmissions. Figure 52 compares error correction performance of ARQ and HARQ methods.

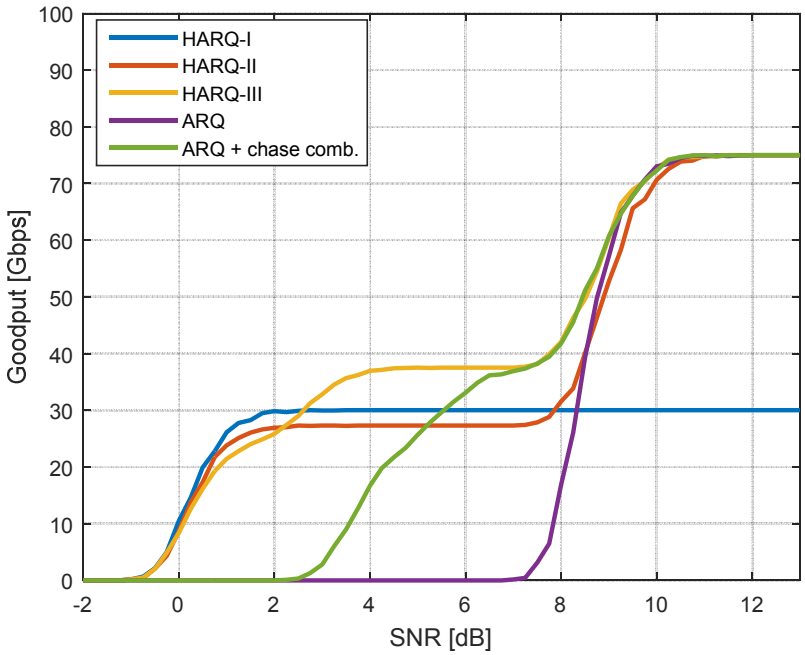


Figure 52: Comparison of HARQ methods. HARQ-III achieves the highest average goodput. 100 Gbps PHY throughput is considered, the simulation includes overhead of PHY preambles and RF-turnaround time.

The most complicated HARQ-III method achieves the highest average goodput. This approach is efficient in channels with both low and high SNR. The simplest ARQ and HARQ-I provides the lowest goodput. HARQ-I is inefficient in high SNR communication channels, as ARQ for low SNR links.
HARQ-II and HARQ-III algorithms send initial transmissions without FEC coding (without redundancy). If the SNR is lower than 8 dB (according to Figure 52), then the first transmission has very little chance to be successful, and retransmission procedure is used in most cases. If 20-30% of frames are lost, then it is better to increase FEC redundancy for the initial transmission. This simple observation can be used for link adaptation methods [93], [95]–[97]. The amount of redundancy can be adopted according to the quality of the communication channel (SNR). If most of the frames get lost during transmission, the amount of redundancy data has to be increased. If all frames are decoded successfully, then the amount of redundancy data can be reduced or even omitted. Results of this method are shown in Figure 53. When the link adaptation method is enabled, the simplest HARQ type-I is almost as efficient as HARQ type-III. It means that the simplest method can replace the most

complex solution in practical implementation. This significantly reduces complexity of a 100 Gbps data link layer processor, and is discussed in sections 3.11-3.12.

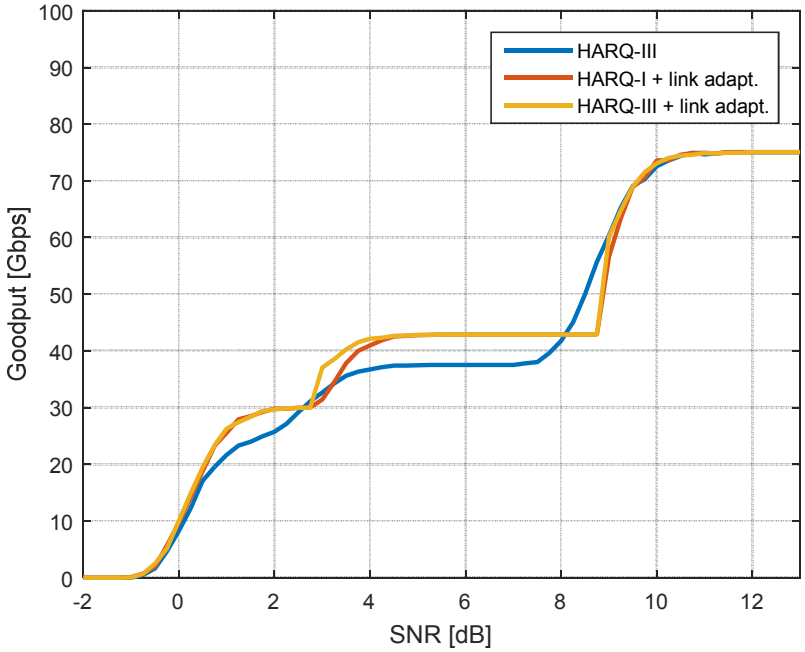


Figure 53: Comparison of link adaptation and HARQ-combined methods. With activated link adaptation, HARQ-I achieves goodput comparable to HARQ-III. However, HARQ-I processing effort is significantly lower when compared to HARQ-III. 100 Gbps PHY throughput is considered, and the simulation includes overhead of PHY preambles and RF-turnaround time.

2.6 High speed serial transceivers

Serial transceivers allow interfacing external peripherals to ASIC/FPGA chips (e.g., optical SFP modules, array of SMA connectors, PCI-express lines). Such transceivers use four IO pins and communicate in full-duplex mode reaching transmission speed of 32.75 Gbps on the newest Virtex UltraScale devices [98]–[102], [103]. More often used are slower GTH/GTX transceivers with goodput of ~12 Gbps. Virtex7 FPGA usually supports 36 – 72 of such transceivers in parallel [104]. This is enough to transfer 100 Gbps streams to and from an FPGA device. Such communication is not possible using general-purpose I/O pins. Thus, the transceivers play an essential role in prototyping high-speed technologies. ASIC variants of the GTH/GTX interfaces are available as well, but due to licensing restrictions, only Xilinx library elements are discussed in this work.
Figure 54 presents a simplified block diagram of a typical GTH/GTX transceiver configured for 10GBASE-SR operation (10 Gbps Ethernet [101], [102]). The

Ethernet (10GBASE-SR) is a standardized and widely used protocol for optical fiber networks. It is one of the basic interfaces supported by high-performance computers (e.g., Tilera cards[13] [105] and IBM Bladecenters [106]). Therefore, an FPGA device supporting the high-speed serial transceivers can be easily connected to those computers. The serial transceiver supports a 64-bit bus at 156.25 MHz clock [101]. The design implemented in the FPGA/ASIC has to be compatible with the provided bus.

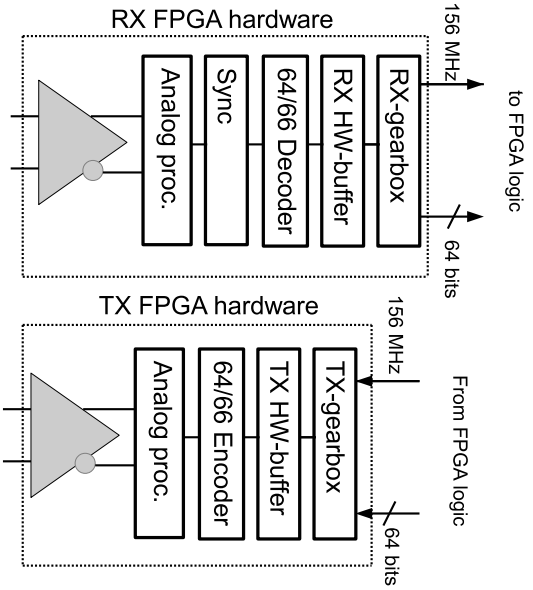


Figure 54: Simplified block diagram of a single GTX/GTH serial transceiver configured for 10GBase-SR Ethernet operation (10 Gbps). The transceiver's interface requires 64-bit data buses and two clocks (tx_clk = rx_clk = 156.25 MHz).

2.7 High speed wireless DLL implementations

The implementation presented in this work is currently one of the fastest data link layer (DLL) processors in the world. Even if research in 100 Gbps wireless communication systems is currently a very hot topic in the scientific community, there are not many articles focused on the DLL published yet. Researchers mostly concentrate on the PHY-layer, but not on the DLL processing (e.g., DFG-SPP1655 [10] and results published in [4]). Thus, the number of published articles in the targeted area is very limited. Currently, there is only one alternative paper about

[13] Tilera card is shown in Figure 149 and Figure 150. It is a multicore processor manufactured by Tilera. It consists of a mesh network of 72 '*tiles*', where each '*tile*' consist of a general-purpose processor, cache, and a non-blocking router, which is used to communicate with the other *'tiles'* on the processor.

research in the 100 Gbps wireless-DLLs [21]. The author considers a HARQ approach for nanonetworks operating in 300 GHz band with OOK (On/Off Keying) modulation. The presented simulation models uses Hamming(15,11) channel coding together with ARQ. This uncomplicated solution is considered due to millimeter distances used in the targeted application and it is not a recommended approach for general purpose 100 Gbps transceivers[14]. The FEC engine presented in this work is more universal and can use the redundancy data more effectively in contrast to the mentioned solution. Additionally, here not only simulation results are presented, but also a fully operational FPGA demonstrator and results of ASIC synthesis.

Another interesting solution solving the problem of high speed DLL processing is shown in [45]. The achieved data rate of 1.3 Gbps is significantly below the target of this work, but a similar technology is used, and all DLL aspects are deeply explained. Additionally, the authors have built a powerful demonstrator that is examined in a machine vision system. The demonstrator uses frame aggregation and concatenated convolutional coding together with RS(255, 239). The convolutional decoder uses a Viterbi algorithm with 5-bit quantization (soft decision decoder) [107]. In contrast with [45], the solution presented here operates with much higher data rate, and is highly parallelized. Additionally, link adaptation methods and iterative turbo product codes (TPC) are used instead of the static Viterbi-RS decoder.

The fastest communication systems available in consumer electronic market are also significantly slower that the solution presented in this thesis. The state of the art LTE modems transfer up to 0.3 Gbps [81]. The planned successor (LTE-A) standardizes the maximal throughput up to 1 Gbps [108]. The WLAN 802.11ad (60 GHz [7]) supports transfers up to (theoretically) 6756.75 Mbps [5] and uses link adaptation with LDPC coding (1/2, 5/8, 3/4, 13/16). Even if LTE and 802.11ad implementations do not support the required data rate, both standards use relatively complicated DLLs. The complexity level is higher compared to the solution presented here, due to multiple accesses to the medium, complex HARQ schemes, and very efficient soft decision LDPC coding.

[14] Length of Hamming(15,11) code word is too short and the redundancy data is used inefficiently. This is investigated in section 3.15.3, and is proved by simulations shown in Figure 121. Moreover, the proposed Hamming decoding is a simple single pass algorithm and is usually less efficient than iterative solutions proposed in sections 2.4.7 and 3.16.

3. Searching optimal architecture for 100 Gbps data link layer processor

This chapter presents simulation results of the designed 100 Gbps data link layer processor. The goal is to find an optimal architecture for a 100 Gbps DLL system and a mapping to hardware implementation (FPGA/ASIC). The simulated DLL models use real PHY parameters taken from 60 GHz 802.11ad WLAN standard [7]. Applicability to 100 Gbps systems at a frequency level of 240 GHz is assumed even if it cannot be proven in all aspects. Additionally, the chapter compares error correction performance of common FEC algorithms. Furthermore, an improved turbo product codes (TPC) approach is presented. The solution achieves higher error correction performance than the state of the art TPC. Firstly, the activities of the DFG-SP1655 research group, challenges of the wireless 100 Gbps data link layer design, and the concept of lane processing are explained.

3.1 Architecture of the investigated system

This work is related to End2End100 project and cooperates with other proposed projects of the DFG Special Priority Program 1655 (SPP1655) on "Wireless 100 Gbps and beyond", e.g., the Real100G.COM and Real100G.RF [10]. This group of projects investigates a complete wireless 100 Gbps system at ultra-high frequencies (250 - 330 GHz). In the End2End100 project, the main technical idea is to investigate an innovative concept for a Network Interface Card (NIC) working at 100 Gbps wirelessly. Figure 55 shows a layered functional diagram as investigated in the DFG projects.

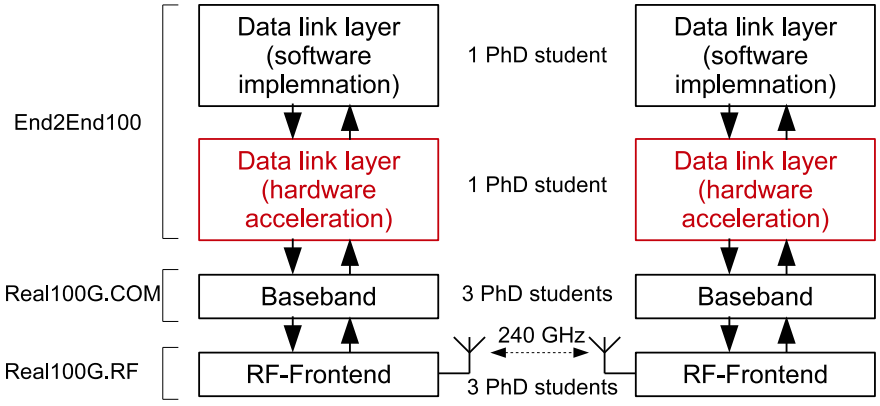


Figure 55: Architecture of a complete 100 Gbps transceiver investigated in the SPP1655 DFG projects. Design and implementation of the data link layer hardware accelerator marked in red is a part of this work.

The data link layer processing is divided into two parts: Tilera many-core processing[15] (software implementation) [105] and hardware acceleration (FPGA/ASIC implementation). Design and implementation of the hardware accelerator (marked in red in Figure 55) is the main contribution of this work. The Tilera software and hardware accelerator are parts of the End2End100 project. The software implementation (Tilera many-core processor) assembles and disassembles data frames, schedules retransmissions, and divides data into fragments (selective fragment retransmission [109]). The hardware accelerator (FPGA/ASIC) calculates checksums (CRC), forward error correction redundancy (FEC), performs link adaptation, and aggregates data frames. Optionally, ACK-frames can be automatically generated by the hardware logic without involving the software. The hardware accelerator operates fully independently and transparently for the software processors.
The baseband is a mixed signal processor designed under Real100G.COM project. The main functionality supported by the implementation is parallel sequence spread spectrum (PSSS[16]) processing and channel deconvolution [12]. The RF-frontend is implemented under the Real100G.RF project.
For more detail on proposed architecture and employed hardware, see sections 4.1 and 4.3.

3.2 Challenges of the wireless 100 Gbps data link layer

This section discusses major challenges of designing data link layer for 100 Gbps wireless communication.

3.2.1 Challenge 1: Ultra short processing time

To achieve 100 Gbps data transmission, a single frame of 1500 octets has to be processed within 120 ns (the frame size corresponds to the maximal Ethernet payload size [110]). Consequently, a high-end ASIC running at 1 GHz needs 120 clock cycles to process a single frame. Therefore, this ASIC has to process 13 bytes of the frame in each clock cycle. This processing includes forward error correction (FEC), cyclic redundancy check (CRC), acknowledge frames (ACK) generation, frames aggregation, and writing data to memory. A single Viterbi decoder at 1 GHz requires approximately 12100 ns to process the frame [50], but as mentioned before, the complete processing has to be finished in approx. 100 times shorter period. It clearly shows that sequential processing cannot support ultra-high-speed wireless transmissions, and all processing algorithms have to be selected carefully for the targeted data rate, even if one of the fastest CMOS technologies is considered.

[15] Tilera card is shown in Figure 149 and Figure 150. The Tilera is the multicore processor manufactured by Tilera. It consists of a mesh network of 72 '*tiles*', where each '*tile'* consist of a general-purpose processor, cache, and a non-blocking router, which is used to communicate with the other *'tiles'* on the processor.
[16] PSSS processing is explained in subsection 1.1.8.

3.2.2 Challenge 2: Bit errors and Forward Error Correction

Today, there are various implementations that support 100 Gbps in wired networks, for example Ethernet 802.3ba based on optical fiber cables [3] running on Altera FPGA platform [111]. In theory, these high-speed implementations might be used, with some adaptations, as the data link layer of 100 Gbps wireless systems. However, it will work inefficiently because the data link layer of wireless systems has to cope with unpredictable bit error rates (BER), leading to more complex solutions [112]. The BER in wireless communication can vary by several orders of magnitude. For example, the BER of high-speed wireless RF frontends presented in [4], achieves BER in range 1e-10 to 4e-3. Therefore, the FEC has to be adopted to the channel conditions and frame fragmentation has to be applied [113]. This increases the average code rate and compensates the unpredictable BER on wireless links. To clarify the issue, a statistical frame error rate calculated for BER=1e-3 is shown in Figure 56 (the selected BER corresponds to a value achieved by 100 Gbps wireless RF-frontend presented in [30]). In the presented case, frames longer than ~0.6 kB cannot be successfully delivered without FEC and fragmentation.

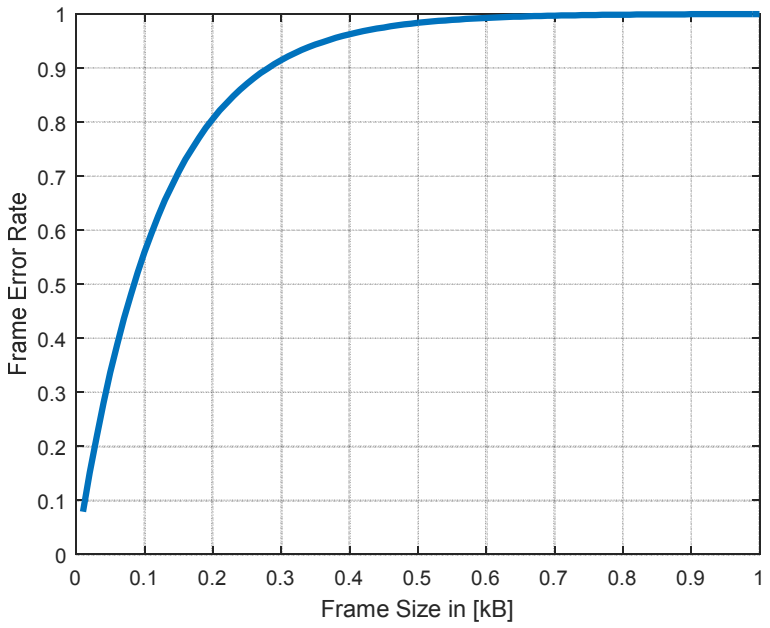


Figure 56: Frame error rate as a function of frame size.

Additional difference between the optical and wireless communication is duplex switching. Optical communication can use two separated fibers to perform uplink and downlink transmissions. Wireless transceivers are limited in this aspect. In most cases, half-duplex communication takes place. A radio can be in either receive or transmit mode, but usually never in both states at the same time considering a typical RF-transceiver. There are some possibilities to use frequency or code multiple access

(e.g., full-duplex radio is presented in [114], [115]), but for very fast links those solutions consume too much bandwidth and are too complicated to manufacture. Thus, in this work, the state of the art half-duplex radio is considered. This assumption reduces the maximum data transfer on the link. Additionally, some of the communication time is wasted because of the PHY switching between receiving (RX) and transmitting (TX) modes[17]. Due to these factors, the data link layer for the wireless 100 Gbps communication has to be considered as new research.

3.2.3 Challenge 3: FEC redundancy data size

The next aspect to consider is the optimal redundancy data size (link adaptation [93]). On one hand, great redundancy data size limits user data significantly and makes the frame inefficient. On the other hand, little redundancy data size increases the probability of non-corrected errors after the FEC. The optimal ratio depends on the BER. The goal is to add exactly as much redundancy as it is required.

3.2.4 Challenge 4: Memory latency

Transceivers need memory to buffer incoming and outgoing frames. Some additional memory resources can be necessary for fragmentation and aggregation on both sides. An Ethernet frame with size in range of 64 – 1518 bytes has to be processed within 5 - 120 ns. Therefore, processing shortest Ethernet frame requires memory with latency << 5 ns. Additionally, at 100 Gbps data rate, the transceivers need 12.5 GB of memory to store the transmitted data over the last second. State of the art computers and FPGA kits have few GBs of DDR3 or DDR4 memory available. However, the access time to such memory is too slow for 100 Gbps processing. The estimated access time for DDR3 memory is around 45 ns [2], but as mentioned before, << 5 ns is required. To overcome this problem, the proposed design avoids buffering of big data chunks, so memory usage is minimized.

3.2.5 Challenge 5: Interfaces

A platform for prototyping systems that require massive data flow and packet processing such as 100 Gbps has to be equipped with the fastest interfaces available on the market. For the End2End project, some high-end FPGA platforms are considered. The state of the art Virtex7 VC707 and VC709 development kits do not support any interface with the required 100 Gbps data rate [100], [99]. The recently released VCU108 Virtex UltraScale has a possibility to prototype the 100G Ethernet, and that is only one solution with standardized 100G interface from Xilinx [116]. However, the board is do not support PCIe 3.0 x16 port to exchange 100G streams

[17] Switching the RF frontend cost some time, and during this time user data is not transmitted. Therefore, the overall transmission goodput is reduced (as explained in subsection 1.1.4).

with a PC system. Thus, the card cannot be used to demonstrate a typical network interface card (NIC) supporting the data rate.
One of industrial suppliers [117] has recently presented a dedicated platform for 100 Gbps development. The solution uses an FPGA accelerator based on a Virtex7 chip. The system supports 100G Ethernet and communicates with a host processor by a custom-designed PCIe 3.0 x16 port. Even if such solution exists, it is rather a very specialized platform for networking, and it is still a challenge to find a hardware for the 100 Gbps data link layer demonstrator.

3.2.6 Challenge 6: Forward error correction complexity

Forward error correction (FEC) allows reducing the effective BER on the data link layer. This significantly improves robustness and goodput of the system, but the FEC gain is very expensive in terms of the processing effort. In [1] logic area consumed by Reed-Solomon, Viterbi, and LDPC decoders is introduced. The ½-rate Viterbi decoder with 5 bit soft coding implemented in the IHP institute [107] requires logic area of approx. 23 Virtex7 FPGAs to deal with the targeted 100G stream. Additionally, subsection 2.4.11 compares Matlab implementation of FEC coders. The state of the art RS(255,239) coding achieves only ~3.8 Mbps when run on a modern PC processor. Thus, approx. 26000 processor cores would be required to support FEC calculation for the targeted goodput. This clearly shows that FEC implementation for 100 Gbps networks is an extremely demanding task.

3.2.7 Challenge 7: Power consumption

In [2] a data link layer implementation for 100 Gbps Ethernet is presented. The solution uses state of the art Intel Xeon processors and pure software implementation. At that time, it was dissipating up to 650 Watts of power. Furthermore, wireless equivalent of this implementation could require even more energy, due to a higher and more unpredictable BER on wireless channels.
The energy consumption of a complete 100 Gbps wireless solution has to be reduced to around 10 pJ per transmitted and processed data bit. That gives around 1 W of consumed power for the whole implementation, including the RF-frontend, baseband, and data link layer processor. Currently, the proposed implementation is nowhere near this limit. Serial links used for interconnection of the data link layer demonstrator alone consume more than 10 W of power [102].

3.3 Lane processing concept

The main technical idea in End2End100 project is parallel lanes processing concept. It is difficult to handle the data rate of 100 Gbps in a single processing pipeline. Thus, the data is split into parallel lanes (Figure 57 and Figure 58). A similar lane approach is employed for 40 Gbps and 100 Gbps Ethernet PHY [118]. In

End2End100 project, the lanes are proposed for data link layer processing, especially for FEC calculations. Such a solution parallels the processing. From point of view of implementation, *n*-lanes reduce the required clock speed *n*-times.

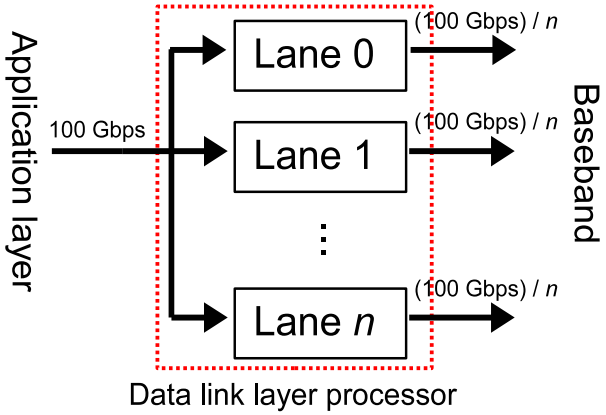


Figure 57: Concept of parallel data link layer processing.

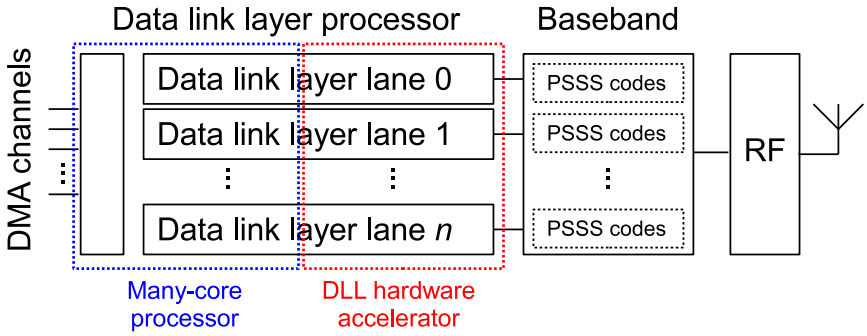


Figure 58: Parallel lanes processing used for the data link layer processor. The data link layer processor consists from the software implementation (marked in blue) and hardware accelerator (marked in red). Both implementations (software and hardware) use a common lane architecture. The lanes are reflected in the PSSS baseband as well.

3.4 DLL simulation model, frame format, and state machine

Figure 59 shows a Matlab simulation of the proposed data link layer processor. TX and RX models use all techniques introduced in the previous chapter: aggregation,

fragmentation, forward error correction (FEC), and a hybrid ARQ with link adaptation.

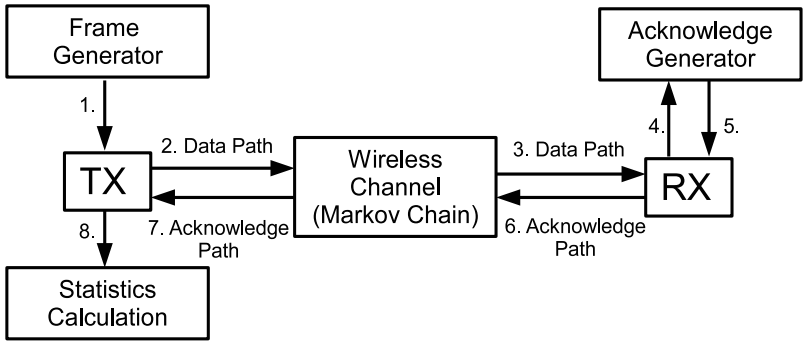


Figure 59: Matlab simulation model. The transmitter sends data frames via an emulated wireless channel to the receiver. The receiver checks the frames, stores sequence numbers of successfully received data frames, and uses the acknowledge generator to transmit an ACK-frame to the transmitter. The transmitter uses the information to retransmit the lost frames.

As a default scenario, a point-to-point communication between a sender (TX) and a receiver (RX) is performed, as depicted in Figure 59. The sender transmits data to the receiver with the data rate of 100 Gbps. Further, the sender waits for a feedback from the receiver to figure out whether the receiver got frames correctly. Details about error recovery are introduced later in this chapter.
To support two-way communication, TDMA (time division multiple access) is applied. In short, the sender stops transmissions of the data frames after a predefined time, and allows the receiver to send acknowledgments.
Figure 60 shows a finite state machine (FSM) that controls the transmission. The transmitter sends a predefined number of frames, and each frame is carrying a predefined number of data-fragments (Figure 61). After that, a single ACK-frame is requested, and a timer is started. If a timeout occurs, then the ACK request frame is retransmitted. This procedure is repeated until the ACK is received successfully. All mentioned parameters of the FSM and frame format are fully adjustable. The communication parameters are deeply investigated in this work.

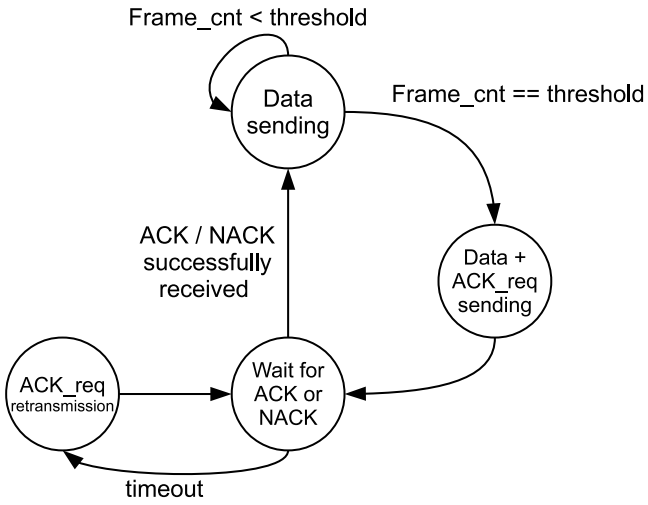


Figure 60: Finite state machine of the transmitter. The transmitter sends a predefined number of data frames. The last frame in the sequence is carrying an ACK-request bit in the frame header. The receiver sends an ACK-frame after receiving the bit. If the data frame with the ACK-request bit or the ACK-frame is lost during transmission, the ACK-request bit is retransmitted in a dedicated frame after a predefined timeout.

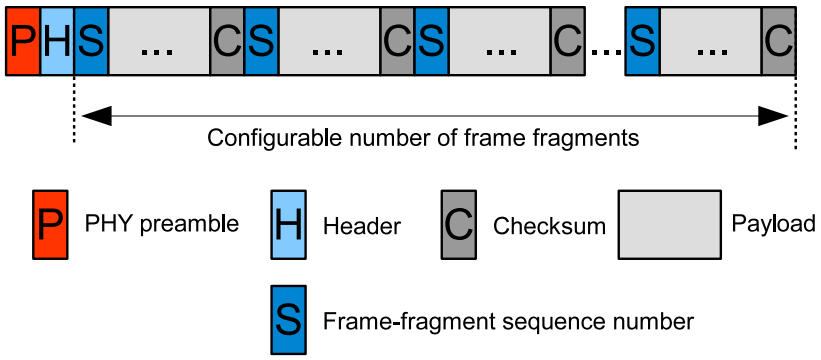


Figure 61: Frame format used in the simulation model. The frame-fragments share a single PHY-preamble and header, but sequence numbers and checksums are distinct. The number of frame-fragments and the length of the fragments are fully configurable. Moreover, the parameters can be adapted to the channel conditions on the fly.

3.5 PHY simulation model

As mentioned in section 3.4, the sender transmits data to the receiver over an unreliable wireless channel. Thus, the wireless channel at the PHY level has to be evaluated. That is, it has to be determined which bits the receiver gets correctly, and which bits suffer from communication errors. Neither the PHY-layer transmission systems, nor the channel parameters are sufficiently know yet. These parameters are concurrently evaluated by associated projects of the DFG-SPP1655. Even more problematic is the lack of characteristic parameters of the analog front-end for the THz band (e.g., power amplifier linearity). It means, the expected error model for the 100 Gbps DLL cannot be defined precisely for now. Therefore, the bit errors of the PHY layer have to be emulated using a generic approach. To simulate bit errors on a wireless channel, a two state Markov Chain can be used (Figure 62).

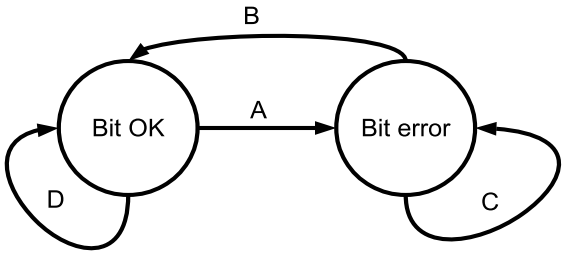


Figure 62: Markov chain employed for PHY bit errors emulation. The state transition probabilities are fully configurable. Thus, the chain emulates an arbitrary selected BER and error characteristics (singles/bursts). Figure adapted by author from [109].

In short, for each received bit the Markov Chain determines if the channel was "good enough" to get the bit correctly. The error correction performance of error recovery techniques depend highly on the type of errors. That is, whether communication problems affect only single bits or a long bit sequence. To emulate various error types, four possible transitions of the Markov chain are considered (transitions A, B, C, and D in Figure 62). For example, the transition 'A' means that after receiving a bit correctly, the next bit will be destroyed. Furthermore, the transition C tells that after not getting a bit correctly, the next bit will also suffer from communication problems. Moreover, in case of burst errors, the model inserts a pseudo-random pattern to the data stream instead of inversing the individual bits. Such a simulation model allows examining various error types and analyzing efficiency of error recovery models. All further simulations presented in this work will use the Markov chain to emulate PHY bit errors.

3.6 Minimal payload size in a single ARQ transmission window

Initially, selective-repeat-ARQ[18] with frame fragmentation and aggregation[19] has been selected for the DLL processing. This solution works efficiently but only if frame and data-fragment sizes are chosen according to the targeted application. In this section, the minimal data size that has to be sent in a single ARQ transmission window[20] is defined. The amount of the payload determines the overall efficiency of the system. If the payload is too short, the PHY switches too often between RX and TX, and goodput is decreased due to idle cycles caused by switching the RF-frontend[21]. The goodput (efficiency) can be estimated by the following formula (3.1):

$$\eta = \frac{t_{data}}{t_{overhead} + t_{data}} \quad (3.1),$$

where:

t_{data} – time used for user payload transmission

$t_{overhead}$ – time used for all other processing: radio switching; preamble, header and CRC transmission.

Very precise timing parameters of the PHY are required for the equation. In this case, 802.11ad [7] PHY timings are used to estimate the initial settings of the DLL. The 802.11ad standard defines the most important timing values: preamble time – 75 ns, and the PHY turnaround time – 1 us. Additionally, the initial DLL frame parameters have to be defined. Thus, length of the header is set to 64 bytes, CRCs – 4 bytes, and the length of the data-fragment-header to 8 bytes. At this step, frame aggregation and fragmentation methods are not considered. Therefore, the calculation does not include an overhead caused by the frame and fragment headers, and retransmissions caused by the channel BER. The estimation is optimistic and defines the upper-boundary of the goodput that can be obtained by the system. To take this optimistic guess into account, the required protocol goodput has to be higher than 99%. The predefined parameters can be used to solve equation (3.1). As a result, the payload size has to be set to at least 2628 kB, and the next power of two value that satisfies the equation is equal to 4096 kB. It means that in a single ARQ transmission window, at least 4 MB of data has to be transmitted, before the radio can be switched to send

[18] In the selected ARQ method an ACK-frame is sent after '*n*' data frames, and all '*n*' frames are acknowledged at the same time. This approach significantly reduces the number of transmitted ACK-frames (PHY-preambles) and RF-turnarounds (For more details, see section 2.3, Figure 15).

[19] See sections 2.1 and 2.2.

[20] The meaning of the ARQ transmission window is defined in Figure 15. The transmission window defines minimal amount of data that is transmitted before the RF-frontends are switched to exchange the ACK-frame (acknowledgment).

[21] This is explained in subsection 1.1.4.

the ACK-frame (acknowledgment). Otherwise, the system will never achieve reasonable goodput. In order to get a better understanding of this issue, Figure 63 shows dependency between the length of the ARQ transmission window and the goodput.

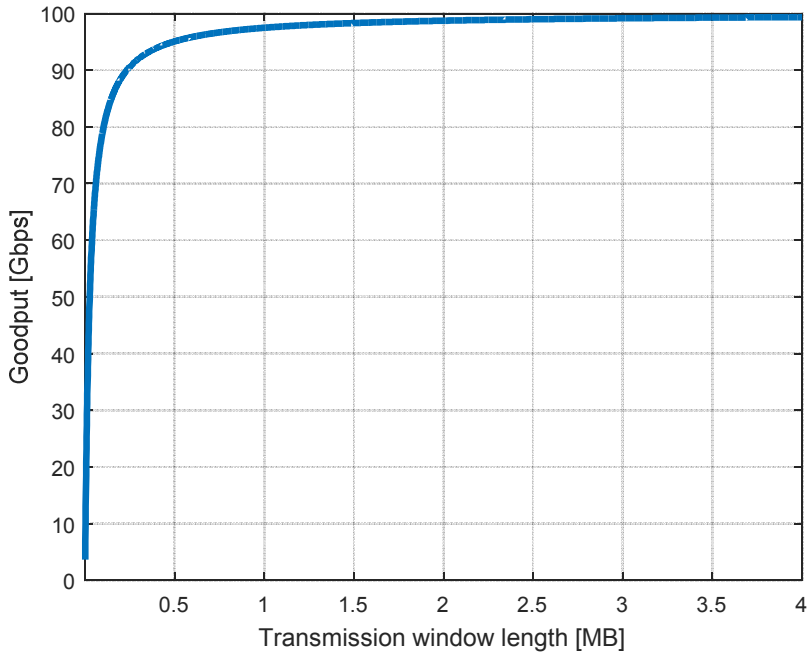


Figure 63: Data link layer goodput as a function of the length of the transmission window. PHY parameters used to generate the results are based on 802.11ad WLAN standard.

3.7 Retransmission fragment length and ACK-frames

If the ARQ retransmission data-fragment size is small, the probability that a large part of the fragmented frame is successfully delivered is increased. Additionally, in case of bit errors in one of the data-fragments, less data has to be retransmitted (selective fragment retransmission is employed). On the other hand, each fragment is equipped with a header, checksum, and is represented by a single bit in the ACK-frame. A large number of small fragments in the data frame induces overhead and requires a long ACK-frame. If the ACK-frame is too long, the frame is vulnerable to bit errors, and some of the ACK-frames can be lost during transmission. This leads to timeouts and retransmissions. Therefore, a compromise between fragmentation and the ACK-frame length has to be found. For example, an ARQ based system shown in Figure 64 might use two fragmentation schemes presented in Figure 65 and in Figure 66 (two and four fragments per data frame respectively). The system with four fragments per data frame achieves higher data transportation efficiency due to

smaller fragment size[22], but also requires transmission of a longer ACK-frame (Figure 65). Each data-fragment requires a bit value and individual addressing in the ACK-frame. Thus, the system with higher fragmentation requires the longer ACK-frame. In the worst case scenario, when many small data-fragments have to be acknowledged, the ACK-frame can be significantly longer than the selected data-fragment length. Therefore, the long ACK-frame is much more sensitive to bit errors than the data-fragments, and the system's efficiency is degraded due to lost ACK-frames. Furthermore, the length of the ACK-frame can be reduced using ACK-frame compression methods (see section 3.9).

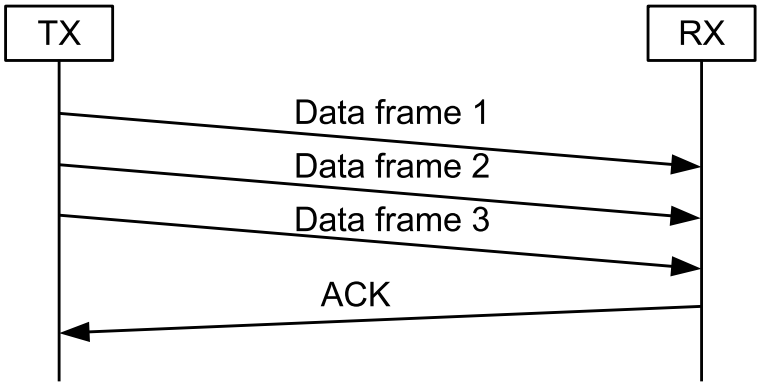


Figure 64: Example of an ACK based system that sends three data frames in a single ARQ transmission window.

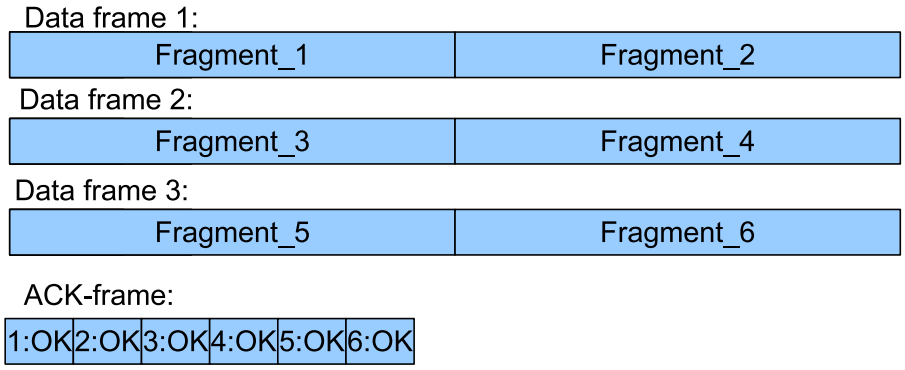


Figure 65: Example of a system that uses two data fragments per frame and corresponding ACK-frame required for the system.

[22] See section 2.1 (frames fragmentation).

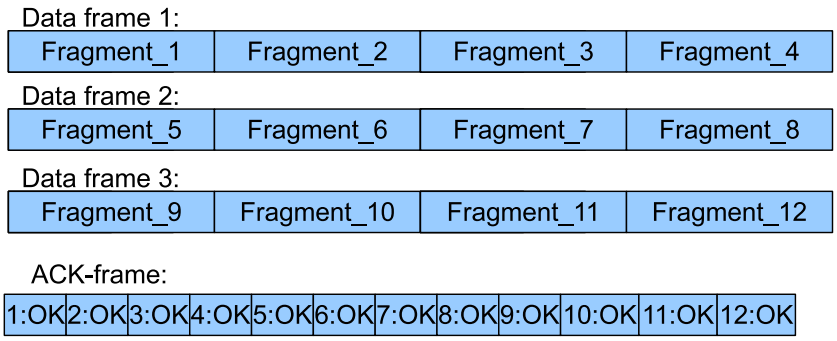


Figure 66: Example of a system that uses four fragments per frame and corresponding ACK-frame required for the system.

Figure 67 shows the relation between the ACK-frame length and the number of the data-fragments in the data frame at constant BER ≈ 1e-3, for the targeted 100 Gbps system. The intersection point (~1 kB) determines the balance of the parameters. It means, frames of length of 4096 kB have to be fragmented to approx. 4096 data-fragments. In such case, a compromise between the ACK-frame length and data-fragmentation threshold is reached. If the fragmentation is increased above this value, then goodput is reduced (Figure 68).

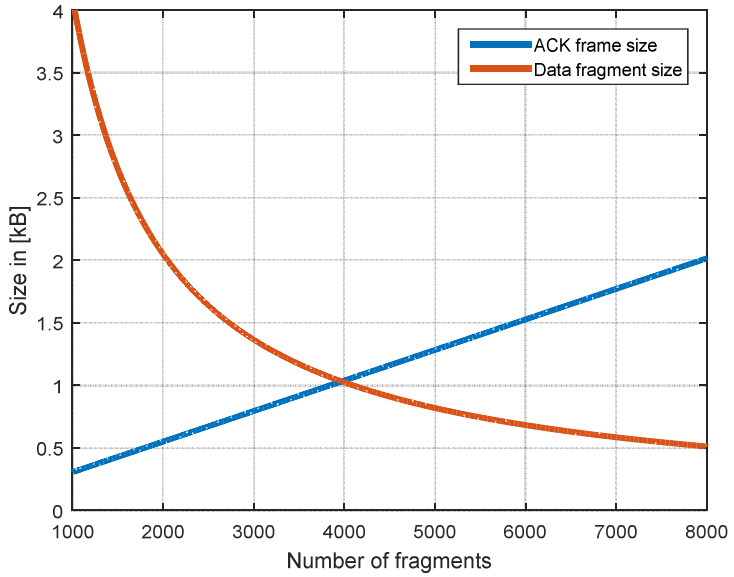


Figure 67: ACK-frame length and data-fragment length as a function of the number of employed data-fragments at BER ≈ 1e-3. The optimal ratio is obtained, when the ACK-frame length is equal to the length of the data-fragments. In the presented case, data fragmentation of ~1 kB is recommended. The value is determined by the intersection point of the two curves.

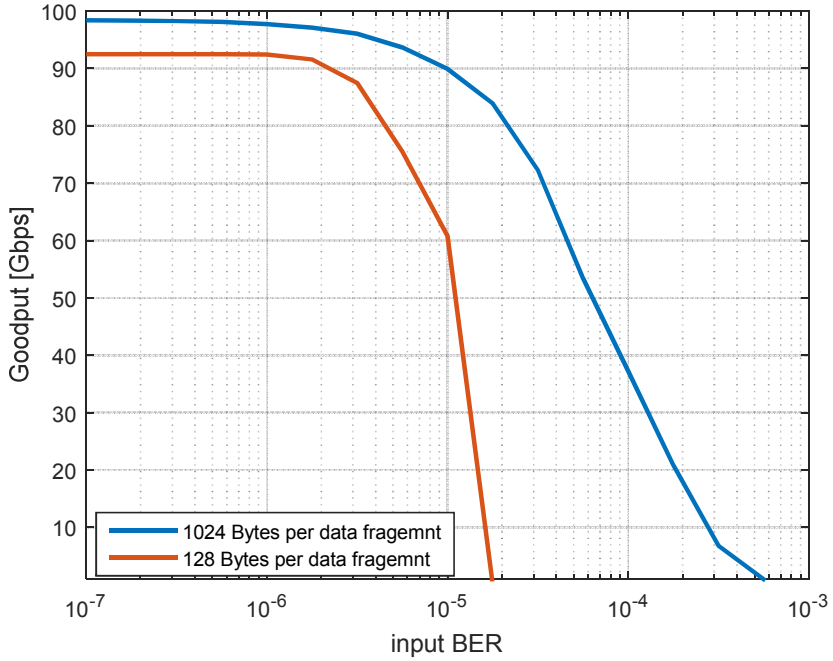


Figure 68: Goodput obtained by two fragmentation schemes. The data-fragment length is set to 1 kB (denoted by the blue curve) and 128 Bytes (denoted by the red curve). In the considered case, the large number of small data-fragments (128 B) induces overhead and reduces the goodput. Moreover, the system with the data-fragment length of 128 bytes requires approx. eight times longer ACK-frame.

3.8 Fragmentation performance as a function of BER

Simulation results presented in the previous section (3.7) do not include BER and FEC coding effects. Each bit error reduces the number of successfully delivered data-fragments, and the optimal fragmentation size (fragmentation threshold) is a function of the BER. Thus, the goodput is strongly correlated with the BER. Additionally, channel coherence time[23] has to be taken into the account, and 4096 kB of payload has to be split into frames. In a default configuration, the system transmits 64 data frames with 64 data-fragments in each frame and achieves 95.49 Gbps at BER = 1e-6 (Figure 69). The coherence time of the channel has to be not shorter than the transmission time of a single frame (5.2 us). Similar requirement is given by the targeted baseband processor, where coherence time has to be significantly longer than 1000 PSSS[24] symbols [11], [119]. This is achievable because the system is

[23] The channel coherence time is commonly defined as the time in which the channel can be considered constant [143].
[24] Construction of a PSSS symbol is explained in subsection 1.1.8.

stationary (a slow-fading channel is assumed). Moreover, multipath propagation is minimized due to high carrier frequency (~240 GHz) and high directivity of employed antennas [120]. Reflections are attenuated due to high path loss of the terahertz band, and high attenuation of the side lobes of the antennas.
It can be observed, that the selected 1 kB-data-fragment size is not optimal for BER higher than 1e-5, and the number of data-fragments should be increased. However, increasing the number of the fragments extends the ACK-frame length, and this leads to losing ACK-frames. Thus, the strategy of decreasing the data-fragment length cannot deal with BER higher that 1e-5. For so high BER values, FEC processing has to be employed. FEC significantly reduces bit errors in the data stream processed by the data link layer (Figure 70). RS(255,237) removes all errors up to BER ≈ 6e-4. FEC with data-fragmentation allows obtaining a goodput of ~81% at BER ≈ 2e-3. For more noisy channels, more powerful coding than RS(255,237) has to be used (this is discussed in sections 3.13, 3.15, and 3.16).

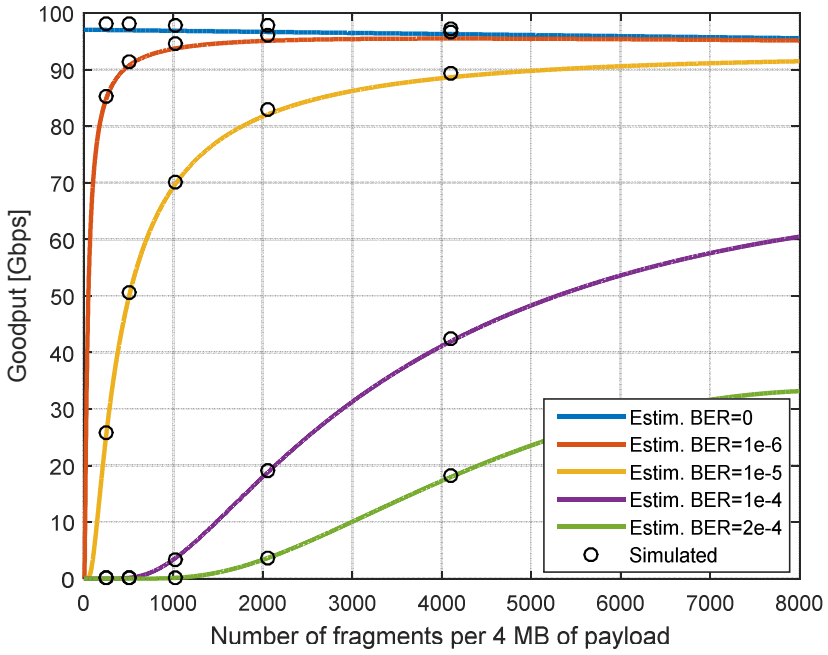


Figure 69: Estimation of the system goodput including BER and fragmentation.

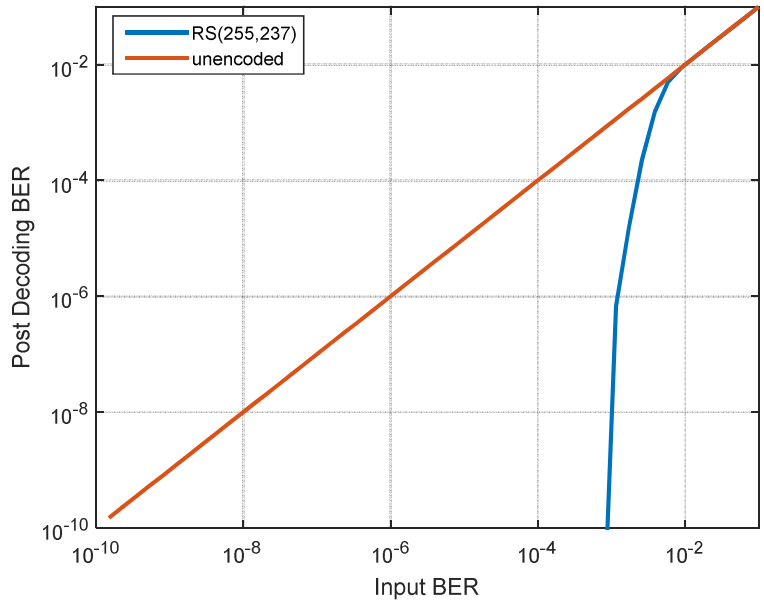


Figure 70: Error correction performance for RS(255,237) coding. The decoder removes all bit errors up to BER ≈ 6e-4.

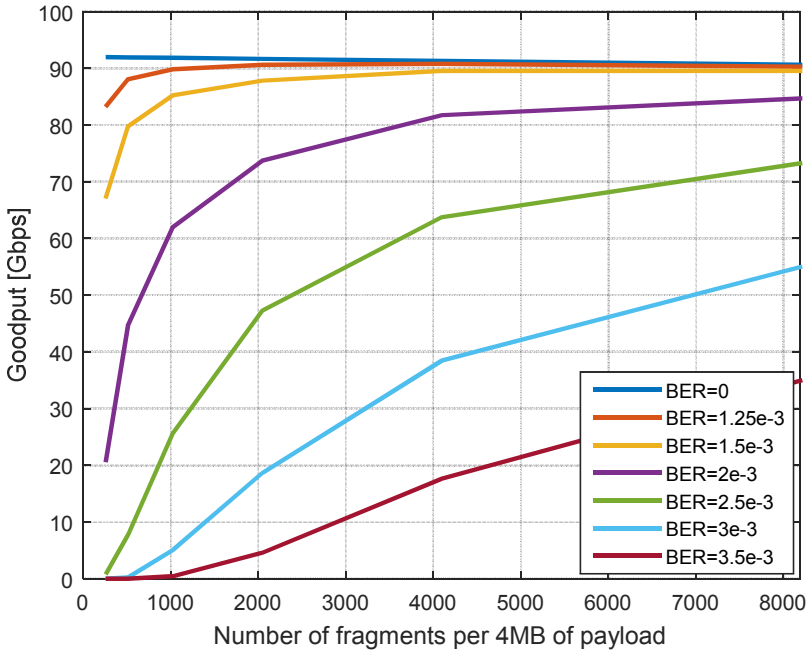


Figure 71: Simulation of the system's goodput including BER, fragmentation, and FEC (RS 255,237).

If FEC is taken into the account, the fragmentation is less sensitive to the input BER (Figure 72). The RS decoding process removes all errors up to input BER ≈ 6e-4. Therefore, in the BER range [0, $6e^{-4}$] the goodput does not vary. However, the goodput drops rapidly if input BER is higher than ~2e-3. This is confirmed by simulations of the system presented in Figure 71 and Figure 72.
For RS(255,237) the best data transportation results are achieved if the RS data size (237 bytes) is equal to the data-fragmentation threshold. In such case, errors from a single RS block influence single data-fragments only. Else, a single faulty RS block influences the neighbor data-fragments. In the targeted 100 Gbps system, such a small fragment size cannot be used, because this leads to a long ACK-frame and overhead induced by fragment-headers and fragment-checksums. Thus, larger data-fragment length is employed (1 kB) and this leads to goodput degradation in the RS encoded system (Figure 73). In the considered case, the proposed fragmentation (1 kB) reduces the maximal input BER for the DLL processor up to 8% comparing with the optimal fragmentation (237 bytes).

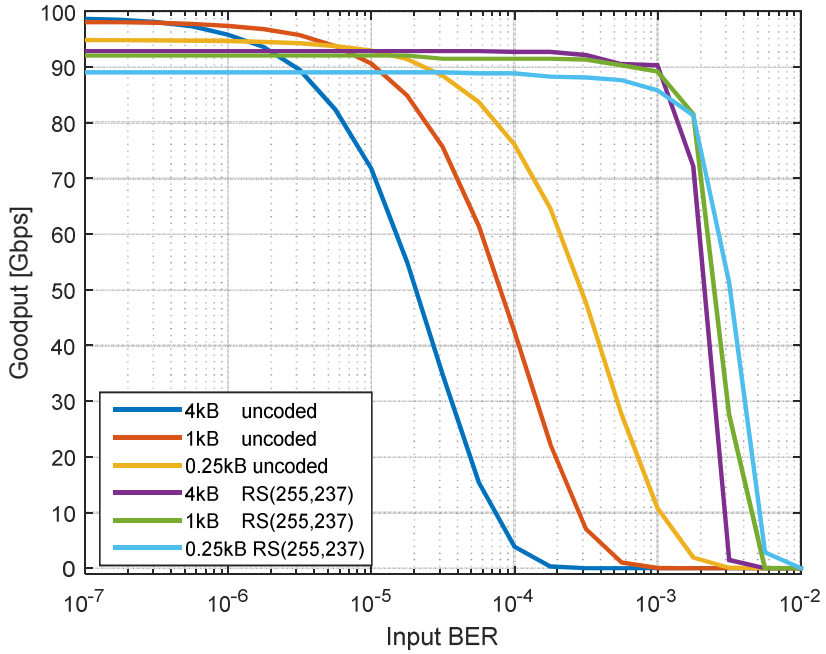


Figure 72: Comparison of three arbitrary selected fragmentation thresholds for uncoded and RS encoded transmissions.

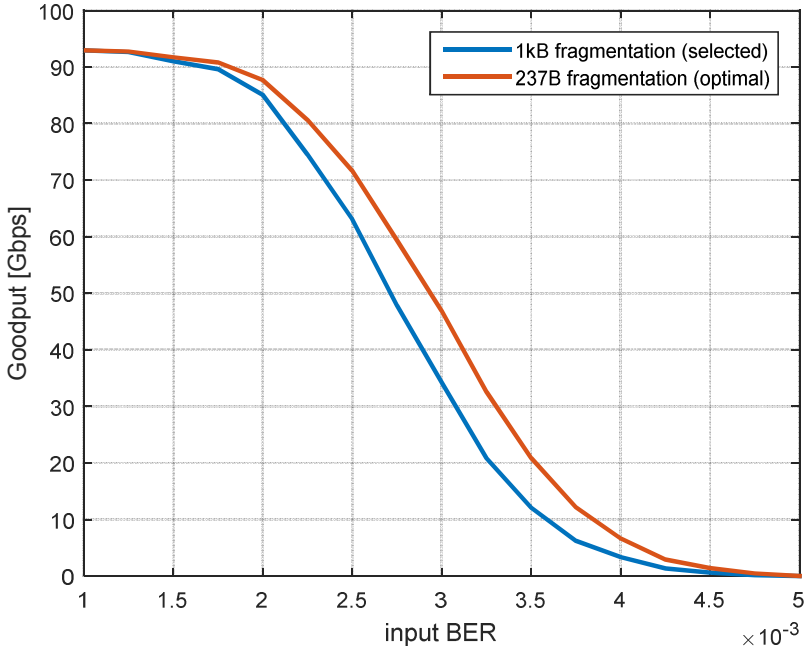


Figure 73: Comparison of goodput for the selected fragmentation scheme (1 kB) and goodput achieved by the optimal fragmentation (237 B) for RS(255,237) coding.

3.9 ACK-frame length and ACK compression

If transmitter and receiver do not exchange information about lost data, some data-fragments might never be delivered, and user payload may lack consistency. To avoid such situations, ACK-frames are sent after a defined number of data frames (after an ARQ transmission window). If the ACK-frame is lost, the transmitter sends an ACK-request frame after a predefined timeout. Additionally, the ACK transmission requires switching the RF-frontends two times (from RX to TX and from TX to RX). During that time, including the timeout duration, user data cannot be transmitted. Thus, ACK-frame retransmissions have a serious impact on the goodput. To avoid the loss of the ACK-frames, and to reduce the number of timeouts, ACK-frames are strongly encoded (e.g., RS 255, 223). This significantly improves the ACK-frame robustness. The overhead induced by the ACK-frame coding is relatively low, because ACK-frames are short (~1 kB) and sent infrequently (1 ACK frame every 64 data frames).

The ACK-frame length depends on the number of successfully received data fragments in a single ARQ transmission window[25] (positive acknowledgment). If the fragmentation threshold is lowered, many small data-frame-fragments have to be sent and acknowledged (as explained in 3.7). Therefore, compression methods can

[25] The meaning of the ARQ transmission window is defined in Figure 15.

be used to reduce the length of the ACK-frames. Figure 74 compares two compression approaches with a basic bit map scheme. A single value and a bit map are sent in the standard bit map scheme. The first value defines the first acknowledged data-fragment number, and the bit map determines all next values. The bit position specifies an offset, and the bit value defines whether the fragment is acknowledged or not. The second and third methods send only a range of addresses of successfully received data-fragments.
Figure 74 compares the maximal ACK-frame length obtained by the proposed techniques as a function of input BER. The proposed method with 15-bit coding significantly reduces the ACK-frame size (< 1 kB) under all input BER conditions. This additionally improves (reduces) frame error rate of the ACK-frames.

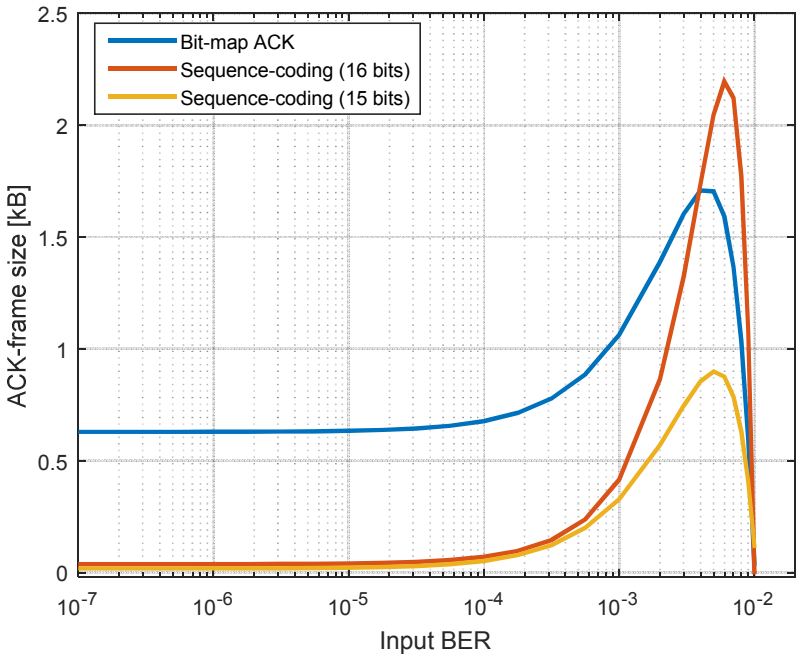


Figure 74: Maximal ACK-frame length as a function of the channel BER.

The sequence coding with 16 bits writes to memory two unsigned integer values (16 bits) that define the first and the last sequence number of successfully received data-fragments in a consecutive subsequence. Such coding is very effective if all or almost all data-fragments are successfully decoded (transmission at BER < 1e-4 according to Figure 74). On the other hand, encoding of a single data-fragment in a long sequence of faulty fragments costs 32 bits – two unsigned integers values (at BER > 1e-3 according to Figure 74). To overcome this problem, a modified coding using the 15 lower bits for a sequence number, and the most significant bit to indicate single values. All three methods are compared in Figure 75 and Figure 76.

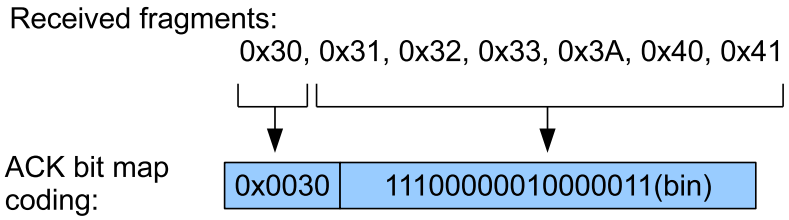


Figure 75: Bitmap-ACK encoding. The first value defines the first acknowledged fragment number (0x0030), and the bit map defines all next values. The bit position defines an offset and the bit value defines whether the corresponding data-fragment is acknowledged (bit set to '1') or not (bit set to '0').

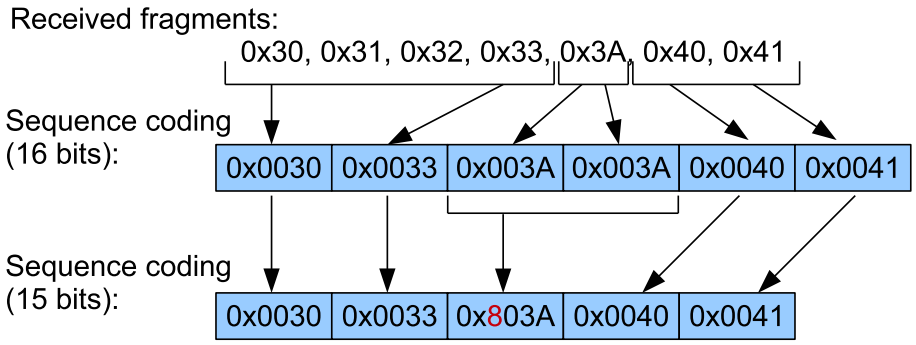


Figure 76: Alternative compressing techniques proposed for an ACK-frame.

3.10 Performance comparison of HARQ-I and HARQ-II methods

State of the art ARQ[26] methods achieve best results for links with very low BER. For the investigated system, ARQ performs efficiently up to BER ≈ 1e-5 (Figure 77). Above this value, the goodput degrades due to retransmissions. Classical ARQ methods do not use any FEC algorithm to fix bit errors in the streams. Thus, in BER range [1e-5, 2e-5] HARQ-II[27] is a more effective solution (according to Figure 77). The method does not retransmit a complete frame, but sends FEC redundancy data for defected frames. It means, data and redundancy information is carried by two separate frames. This requires a doubled number of frame-preambles and frame-headers. Thus, on links with BER higher than ~2e-5 (according to Figure 77), HARQ-I achieves better results. The approach sends data with redundancy in one frame, and the FEC data is always available for FEC decoders. This reduces decoding latency of the data, the overhead induced by frame headers, and does not

[26] ARQ processing is explained in section 2.3.
[27] HARQ methods are explained in section 2.5.

require cache memory. However, for links with BER < 1e-5 the unnecessary redundancy reduces the overall transmission efficiency (goodput and energy efficiency).
HARQ-III is not considered in this section due to the complexity of the approach. HARQ-III requires a punctured FEC code (or a FEC code with erasure indications) and memory for defected frame fragments (see also section 3.11).

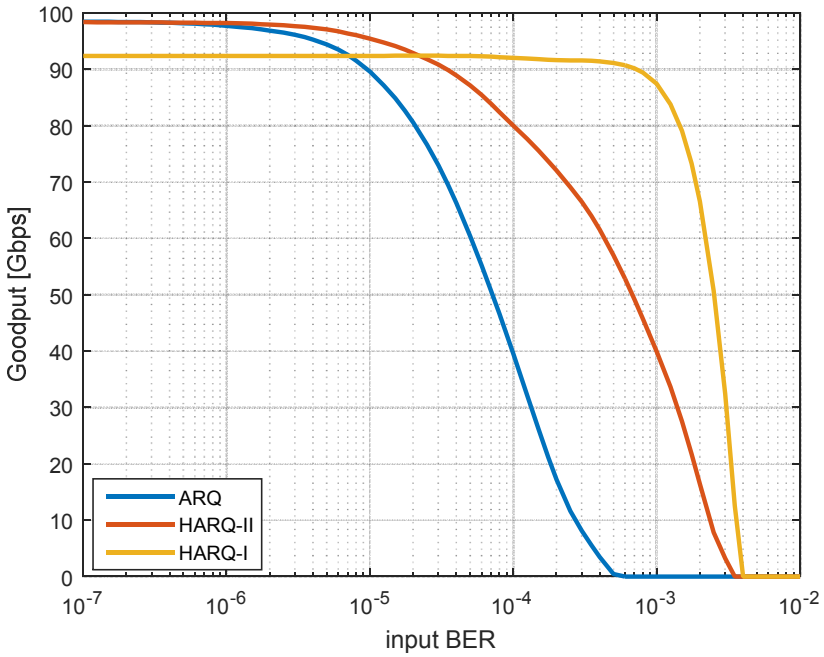


Figure 77: Comparison of goodput obtained using ARQ, HARQ-I, and HARQ-II. ARQ and HARQ-II performs well on links with 'low' BER (in the considered case BER < 1e-5). HARQ-I achieves good results for links with 'high' BER (in the considered case BER > 2e-5).

3.11 HARQ-II memory usage

If HARQ-II (or HARQ-III) is in use, some memory is necessary to store defected frames. The frames await forward error correction data and have to be stored in a fast cache memory. If the channel condition is getting worse, several frames have to be buffered (Figure 78). The maximum memory usage occurs when all frames in the HARQ transmission window are defected. If the protocol do not prioritize retransmissions of the defected data over new data transfers, the cache memory usage grow infinitely. Thus, the proposed DLL protocol prioritizes repetitions of the defected frames over new data transfers, and the memory usage is limited to double size of the ARQ transmission window (2×4 MB, according to section 3.6). Furthermore, if the ARQ transmission window has to be extended due to poor RF-

transceiver turnaround time, then also the memory overhead will be higher. Although, the buffer size of ~8 MB is not an issue for modern FPGAs/ASICs, the required access time << 5 ns and the speed of ≥100 Gbps is a serious challenge for the memory subsystem [35]. Standard DDR3 modules cannot be used for this purpose [2]. Moreover, HARQ-II (and HARQ-III) requires fast searching algorithms to match the defected frames with received FEC data. Due to this reasons, HARQ-I method is preferred for hardware implementations instead of HARQ-II (and HARQ-III). HARQ-III method is even more complex than HARQ-II and requires even more memory accesses and searching operations in the cache memory than HARQ-II. HARQ-I does not require cache memory and searching algorithms for defected frames at all, but is significantly less effective on 'good' RF-links. To overcome this problem, a link adaptation approach can be applied. The link adaptation improves the goodput of HARQ-I approach on 'good' links. This is discussed in section 3.12.

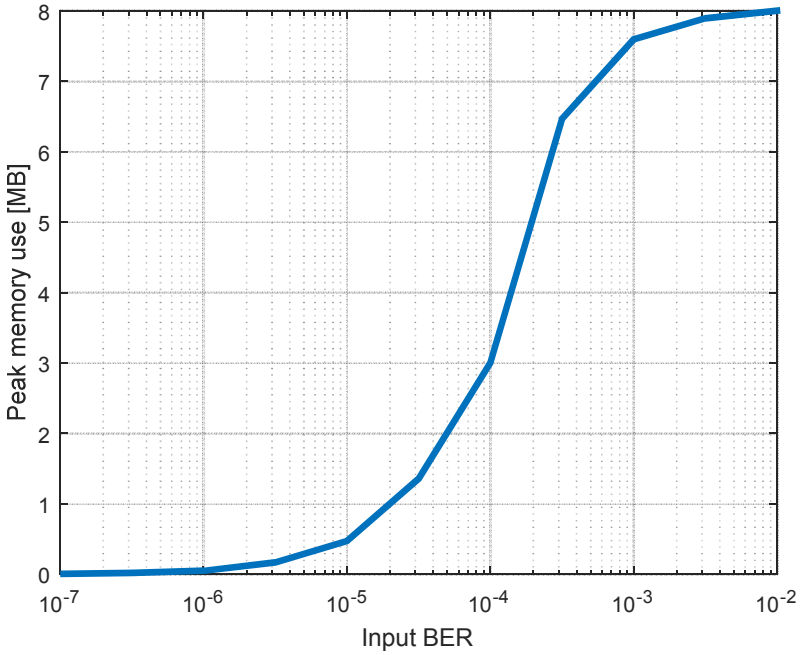


Figure 78: HARQ-II memory usage as a function of input BER.

3.12 Link adaptation

This section describes the concept of link adaptation. This technique significantly improves communication goodput according to channel BER[28]. Together with hybrid-ARQ and frames aggregation, it is one of the most important techniques applied for the proposed DLL processor.

[28] Advantages of link adaptation scheme are explained in section 2.5 (Figure 53).

3.12.1 Concept of link adaptation

It is possible to design a link adaptation algorithm that finds a tradeoff between FEC coding overhead and the demanded error correction performance. The algorithm analyzes the number of successfully delivered data-fragments and the number of corrected errors in the fragments. If the goodput is degraded by losses of data, the FEC coding has to be increased. Such strategy reduces overhead induced by retransmissions on 'bad' links. On the other hand, the algorithm can decrease the FEC coding on 'good' links and reduce the overhead induced by unnecessary redundancy data. Shortly speaking, the approach finds a compromise between FEC overhead and retransmissions, so the goodput of the protocol is maximized under all BER conditions. The most important is to define thresholds, when the FEC code has to be changed. One possible solution is setting the thresholds to the code rates of the employed codes. For example, when RS(255,249), RS(255,239), and RS(255,223) are used, the thresholds can be set to 249/255≈97.6%, 239/255≈93.7%, and 223/255≈87.5%. If the percentage of successfully delivered data fragments is below the given values, then a code with higher correction performance is applied. The thresholds correspond to the code rates and define the upper boundaries of the goodput for the codes. Figure 79 explains the idea of operation of the algorithm. Figure 80 shows the results of the proposed approach that adopts RS coding in four steps: no-coding, RS(255,249), RS(255,239), and RS(255,223).

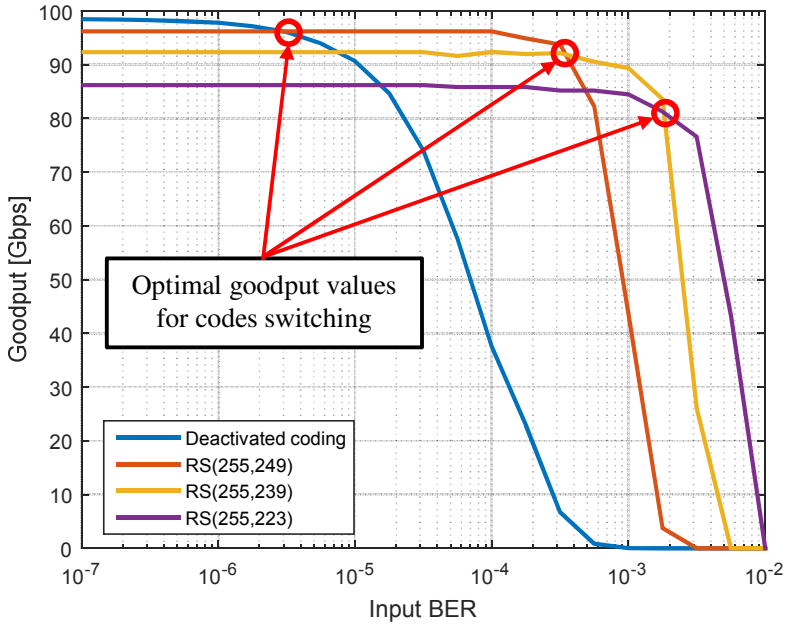


Figure 79: Idea of the link adaptation algorithm. Intersection points of the curves define goodput values when a code switch has to be done.

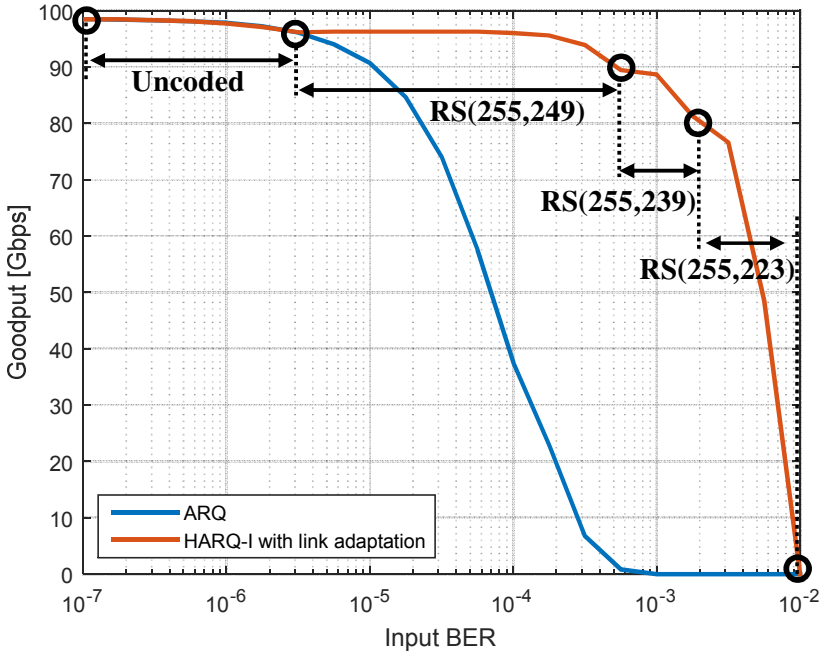


Figure 80: Goodput obtained by employing HARQ-I with link adaptation (denoted as red). The coding is adjusted as function of the input BER. Such approach eliminates the problem of reduced efficiency on 'good' links, when HARQ-I method is applied. In the presented case, the adaption algorithm selects coding from a set of RS codes defined by: [no-coding, RS(255,249), RS(255,239), RS(255,223)].

3.12.2 Proposed algorithm

The approach adopts the redundancy according to the channel BER. When the channel is getting worse, more FEC is added to a frame. The algorithm is discussed according to RS codes but it can be modified for other FEC methods as well. The adaptation routines use the following statistics to decide whether the coding has to be changed: the number of defected data-fragments, the number of all data-fragments, the number of corrected symbols in a RS block, and the number of processed RS blocks (Figure 81). The statistics are represented as hardware counters and are cleared after each HARQ transmission window[29] (~335 us). Firstly, the algorithm selects two alternative codes to the currently used code. One of the codes has higher code rate and another lower. After that, code increase and code decrease branches (Figure 81) decide if one of the alternatively proposed codes might achieve higher goodput than the currently used one. This procedure is run iteratively until all

[29] The meaning of the ARQ/HARQ transmission window is explained in section 2.3 (Figure 15).

supported FEC codes are compared, and the best code is selected according to the recorded statistics over the last transmission window.

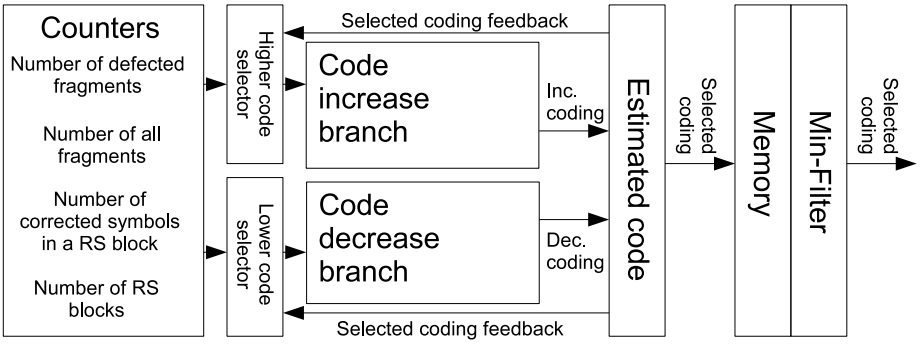


Figure 81: Block diagram of the proposed link adaptation algorithm. The algorithm uses four statistics to decide whether the code has to be switched: the number of defected fragments, the number of all fragments, the number of corrected symbols in RS block, and the number of RS blocks.

The code increase branch (Figure 81) calculates fragment error rate and compares the value with RS overhead. If degradation of the goodput caused by loss of data is greater than the coding overhead, the branch decides to increase the coding. This operation can be represented by the following formula (4.1):

$$\left(\frac{Error\_segments}{All\_segments}\right) > \left(\frac{RS\_overhead}{RS\_block\_size}\right) \qquad (4.1)$$

In the considered case, the *RS_block_size* is constant and equal to 255. To reduce computation complexity, the value can be replaced by 256. It introduces error to the result, but also removes the division from the hardware implementation. Further, a hysteresis is added and the error is masked. The division by 256 can be replaced with binary shift or with routing, so the overhead can be neglected. Therefore, the relation can be represented by the following expression (4.2):

$$\left(\frac{Error\_segments}{All\_segments}\right) > \left(\frac{RS\_overhead}{256}\right) \qquad (4.2)$$

Additionally, formula (4.2) can be simplified. RS codes can correct up to *t* errors, where the *t* is defined by (4.3) [46]:

$$t = \left(\left\lfloor\frac{RS\_overhead}{2}\right\rfloor\right) \qquad (4.3)$$

Then equation (4.2) can be converted into (4.4):

$$\left(\frac{Error\_segments}{All\_segments}\right) > \left(\frac{t}{128}\right) \quad (4.4)$$

This operation reduces the size of variables on the right side by 1-bit, and does not introduce calculation error. Thus, relation (4.4) can be reconstructed into (4.5):

$$Error\_segments > ((t \times all\_segments)/128) \quad (4.5)$$

Additionally, a hysteresis has to be added, and higher coding is switched earlier than indicated by expression (4.5). Nevertheless, the coding is adopted in several steps, and the difference between two adjacent steps is lower than 1% (~2/255). Even if the prediction fails, then the expected goodput degradation is lower than 1%. The hysteresis is constructed in such a way, that the error of the estimation leads to higher redundancy. It means, when the module fails to estimate the value, then transmission robustness is increased, but never reduced. Thus, the effect of the simplification introduced in (4.2) can be neglected. The easiest way to include the hysteresis into formula (4.5), is to replace the division by 128 with division by 256 (4.6):

$$Error\_segments > ((t \times all\_segments)/256) \quad (4.6)$$

Finally, formula (4.6) can be represented in the hardware by the following operations:

'Error_segments' – 16 bit up-counter,
't × all_segmnts' – hardware multiplication (uint4 × uint16),
'/ 256' – binary shift by 8 bits (routing),
'>' – relational operator (16-bits).

The second part of the algorithm, code-decrease branch, can be represented by the following relation (4.7):

$$\left(\frac{Err\_seg\_with\_lower\_code}{All\_segments}\right) < \left(\frac{RS\_overhead}{RS\_block\_size}\right) \quad (4.7)$$

$Err\_seg\_with\_lower\_code$ is a variable that represents erroneous data fragments, which would be decoded incorrectly when the coding would be decreased. The module has to predict this value, and this prediction requires an advanced FSM that continuously analyzes the output of the FEC decoders. Thus, another approach is proposed. Instead of the mentioned statistic, the number of unsuccessfully decoded RS blocks can be used (block error rate). Therefore, it can be assumed that formula (4.8) is more or less correct. Indeed, the value on the right side is equal to the value on the left side only if the data fragment is equal in size to RS data block. In the investigated case, the length of the data block changes with the BER, and it is not possible to fulfill this requirement. Therefore, this simplification introduces some inaccuracy. Furthermore, this inaccuracy is masked by a hysteresis.

$$\left(\frac{Err\_seg\_with\_lower\_code}{All\_segments}\right) \approx \left(\frac{Err\_blocks\_with\_lower\_code}{All\_RS\_blocks}\right) \quad (4.8)$$

$Err\_blocks\_with\_lower\_code$ is a variable that can be calculated by an up counter that is incremented when the following condition is satisfied (4.9):

$$num\_of\_errors > t - 1 \quad (4.9),$$

num_of _errors is a variable usually given by an RS decoder. The value represents the number of defected symbols in the processed RS block. Furthermore, expression (4.2) can be used, and formula (4.7) can be modified to the following form (4.10):

$$\left(\frac{Err\_blocks\_with\_lower\_code}{All\_RS\_blocks}\right) < \left(\frac{t}{128}\right) \quad (4.10)$$

This can be converted into (4.11):

$$Err\_block\_w\_lower\_code \times 128 < t \times All\_RS\_blocks \quad (4.11)$$

Additionally, the hysteresis has to be added. Value 128 on the left side is doubled (4.12):

$$Err\_block\_w\_lower\_code \times 256 < t \times All\_RS\_blocks \quad (4.12)$$

This can be represented by the following FPGA/ASIC operations:

'Err_blocks_w_lower_code' – up-counter
'× 256' – binary shift (routing)
't × All_RS_blocks' – hardware multiplier
'<' – relational operator

At the end of the decision chain, (Figure 81) a memory with min-filter is placed. The filter selects the most robust RS code (with the lowest code rate) that was estimated over the last five HARQ transmission windows. The filter is required to improve adaptation results for channels with coherence time[30] similar to a single HARQ transmission window duration (~335 us, see also subsection 3.12.3 where the adaptation results are investigated as a function of the channel coherence time).
Finally, the algorithm can be represented by up-counters, two hardware-multiplications, and a few relational operators. The results of the multiplications do not exceed 24 bits, so the operations are not an issue for hardware multipliers. The divisions and multiplications by constants in power 2 are realized by FPGA/ASIC routing. Thus, these operations do not introduce any overhead.

[30] The channel coherence time is commonly defined as the time in which the channel can be considered constant [143].

Figure 82 demonstrates the results of the link adaptation algorithm for the targeted data link layer design (validated in VC709 Virtex7 platform). The method significantly improves the overall goodput of HARQ-I approach, especially when the channel BER is low (in the considered case BER < 1e-5). The same method can be used for BCH codes. After a few modifications, the algorithm can be used for LDPC and convolutional codes. The value of the $Err\_blocks\_with\_lower\_code$ variable cannot be estimated for LDPC and convolutional codes so easily like for RS or BCH codes. Thus, the logic responsible for coding decrease branch has to be modified.

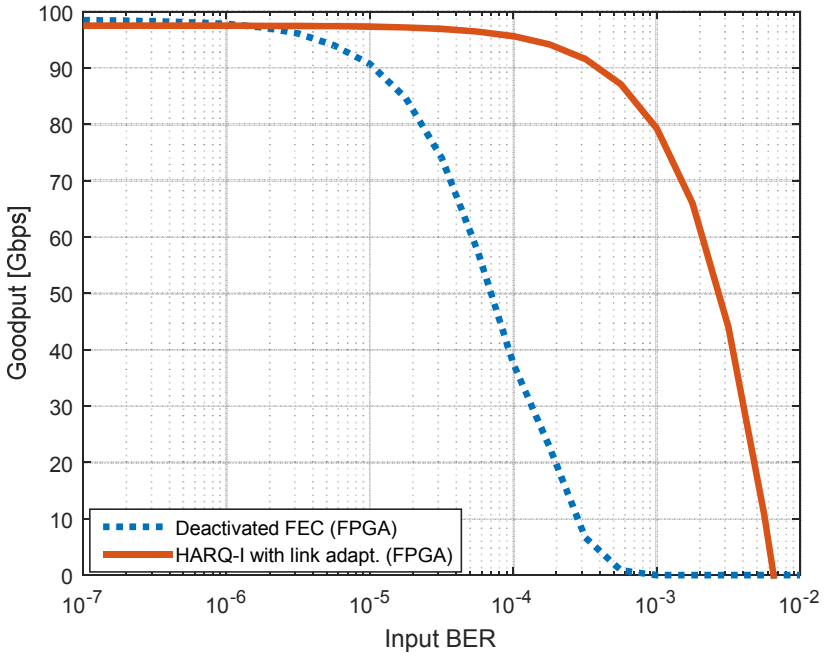


Figure 82: FPGA goodput as a function of the input BER. The link adaptation algorithm is used to optimize the goodput according to the channel BER.

3.12.3 Influence of channel coherence time

The link adaptation algorithm is sensitive to the channel coherence time[31]. If averaging period of the algorithm is too short, the algorithm predicts incorrectly and goodput is degraded compared to static coding. On the other hand, an averaging period which is too long increases response time of the adaptation. Thus, a compromise has to be found, and the algorithm has to be adjusted in the final implementation. In this work, the averaging period is equal to a single HARQ transmission window (~335 us) and filtered by a min-filter (Figure 81). The filter

[31] The channel coherence time is commonly defined as the time in which the channel can be considered constant [143].

selects the most robust coding over the last five HARQ windows (~1675 us). Such an approach improves the robustness of the communication. If an overaggressive coding is selected, the goodput is degraded by redundant data. This leads to goodput degradation by 12.5% in the worst case (1 – 223/255 ≈ 12.5%; the value defined as 223/255 corresponds to the code rate of RS 255,223 coding, which is the most robust RS coding assumed for the link adaptation algorithm). However, when coding is too weak, it leads to a loss of the link between the devices and this reflects in communication blockage (data loss up to 100%). Thus, the algorithm selects the most robust coding that was estimated over the last five transmission windows.
Figure 83 compares the proposed algorithm with static RS coding and compares the results obtained by the algorithm with and without the min-filter. Figure 84 depicts BER characteristic used for the simulations. The algorithm without filtering achieves good performance up to the coherence time equal to the averaging time (335 us). If the coherence time is longer than 335 us, and below 4000 us, the adaptation is not reliable and the goodput is reduced. Filtering significantly improves stability of the algorithm in such conditions, but if coherence time is equal to the filtering period (5 × 335 us = 1675 us), the goodput is reduced. For the coherence time longer than 4000 us, both adaptation strategies work correctly. Generally, the filter period has to be set to a value different from the assumed coherence time. In the other case, the algorithm does not work efficiently.

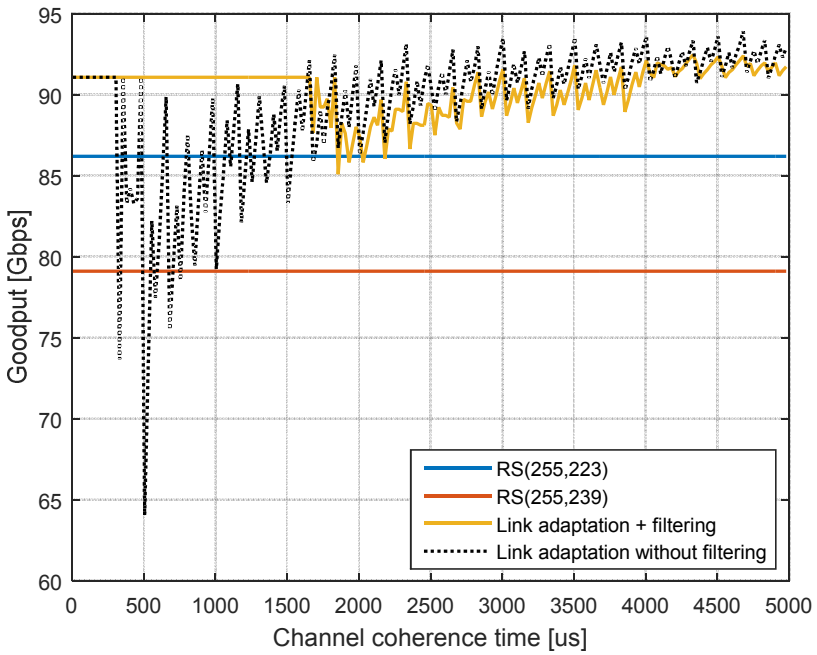


Figure 83: Comparison of goodput obtained by link adaptation algorithm with filtering, link adaptation without filtering, static RS(255,239), and static RS(255,223) as a function of the channel coherence time.

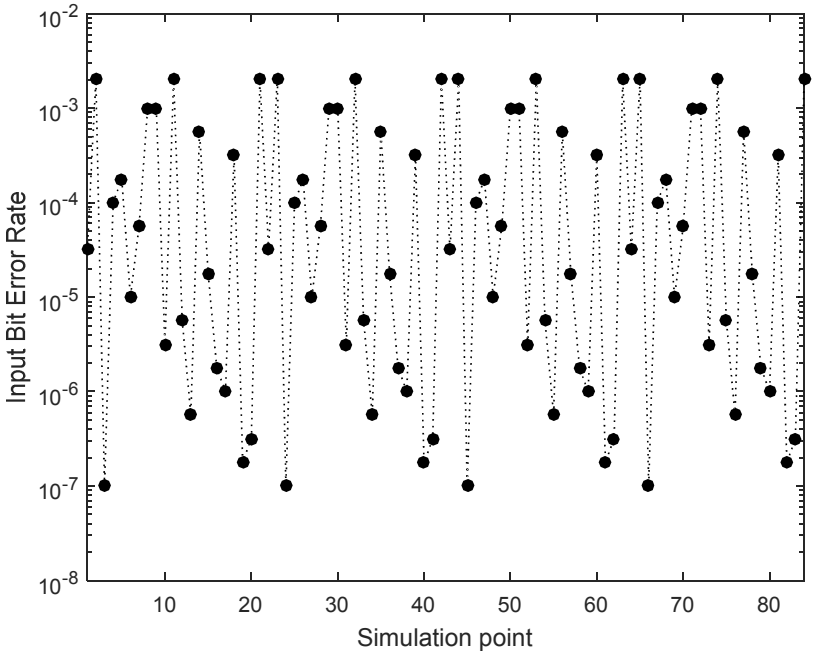


Figure 84: Error characteristic used for testing the link adaptation algorithm. Time difference between adjacent simulation points (coherence time) is configurable in range 5 to 5000 us.

3.13 Error correction performance of RS, BCH, LDPC, and convolutional codes

Section 2.4 introduces RS, BCH, LDPC, and convolutional codes. Moreover, error correction performance of the methods for an AWGN channel[32] is compared (2.4.11). In this section, the methods are tested for the proposed 100 Gbps data link layer processor. A fully implemented Matlab model of the DLL processor[33] is tested against different bit errors that can be generated by the targeted RF-frontend and baseband [11]–[13], [119]. All unknown PHY parameters required for the model are based on 802.11ad WLAN standard. The simulated input BER changes in a wide range from 0 to 1e-2. Additionally, the error length is adjusted for three error characteristics. This allows comparing not only error correction performance as a function of the input BER, but also as a function of the error length (single / burst errors). The Matlab model uses data-frames fragmentation, aggregation, FEC, and HARQ-I concept. The FEC engine supports RS, BCH, HD-LDPC, and convolutional

[32] Additive white Gaussian noise (AWGN) is a basic noise model used in information theory to mimic the effect of many random processes that occur in nature.
[33] See sections 3.4 and 3.5.

coding. All decoders work with hard-decision decoded (HD) input data, and the code rates are set to R ≈ 8/9. The selected code rate allows to construct FEC schemes that supports all 100 Gbps RF-transceivers listed in [4]. According to [4], the input BER for the data link layer is usually lower than 1e-3, and there is no need to use FEC codes with lower code rate. The model allows to simulate expected goodput[34] as a function of the input BER expected on the output of the targeted baseband.

3.13.1 Single errors

Firstly, a simulation for single errors has been performed. In the presented case, RS(255,223), BCH(2047,1816), and HD-LDPC(64800,57600) decoders achieve comparable results (Figure 85). The Viterbi decodable convolutional code (with R=2/3 punctured to R=8/9) obtains poor error correction performance[35]. Thus, the code should not be considered for the targeted implementation. To compare the performance between RS(255,223), BCH(2047,1816), and HD-LDPC(64800,57600), an additional simulation with higher input BER resolution is performed, and the results are presented in a linear scale (Figure 86). In such case, the HD-LDPC code corrects ~15% higher BER than the RS. The BCH decoder provides the best results and corrects ~30% higher BER than the RS. It is important to emphasize the fact that the LDPC and Viterbi decoders use hard-decision decoded input data, and it is an untypical way of using the codes. In most applications, Viterbi and LDPC decoders use soft-decision (SD) decoded bit values.
For BER values lower than 1e-5 uncoded transmission obtains the highest goodput, and in such conditions, the DLL processor has to reduce or disable FEC coding. The processor can do this automatically using link adaptation algorithm discussed in section 3.12.
The simulation (Figure 85) depicts that the selected RS(255,223), BCH(2047,1816), and HD-LDPC(64800,57600) provide similar results in the tested conditions. Thus, decoding complexity has to additionally be considered to choose the optimal coding (see section 3.18).

[34] The goodput metric is explained in subsection 1.1.7.
[35] The presented code with R = 8/9 and hard decodable decoding scheme is very untypical variation of using convolutional codes. In most applications lower code rates (e.g, ½ - ¼) with at least 3-bit soft-decision decoding are used. Moreover, the tested solution uses standard polynomials [23 35 0; 0 5 13] suited for R = 2/3 with constraint length [5, 4] punctured to R = 8/9. For such code rate and hard decision input data, applying RS and BCH codes is preferred instead of using convolutional codes constructed in the presented way.

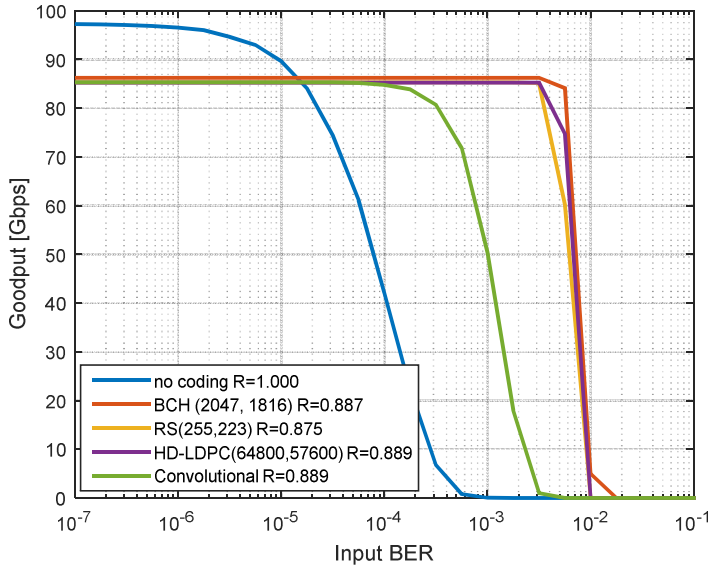


Figure 85: Comparison of FEC algorithms against theoretical error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.00 (single errors). The letter 'R' in the legend denotes code rate of the codes.

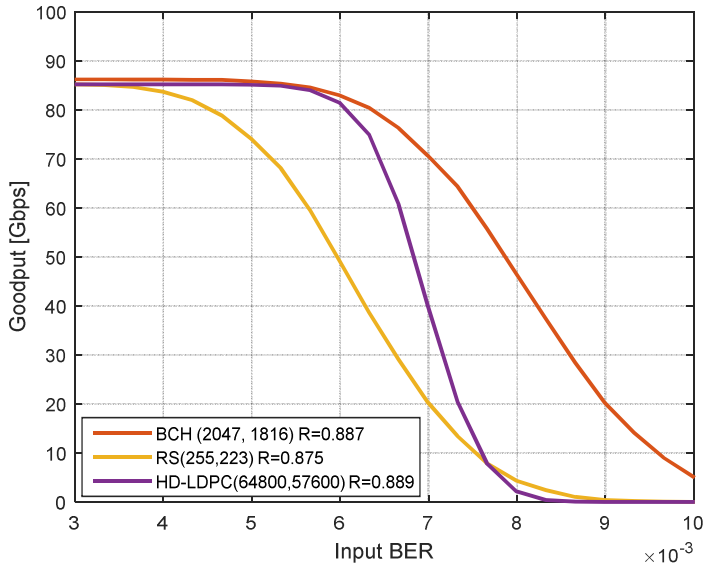


Figure 86: Comparison of FEC algorithms against theoretical error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.00 (single errors). The letter 'R' in the legend denotes code rate of the codes.

3.13.2 Mixed errors

Requirements for the FEC methods are not restricted to high correction efficiency of single errors but also effective correction of burst errors. In the targeted 100 Gbps communication system, a single ripple on the RF-frontend or baseband power supply may disturb decoding of tens consecutive bits (transmission of a single bit at 100 Gbps takes 10 ps). Thus, the system is very sensitive to short time burst errors. In the Real100G.COM project, 15 data bits are packed into a single PSSS symbol[36]. If decoding of a single symbol fails due to noise or synchronization errors, a burst error up to 15 bits is expected on the output of the PSSS baseband. In such case, the data link layer processor has to recover the lost data. Thus, not only the input BER value, but also the characteristic of the error distribution is important. This is reflected in the results of two further simulations shown in Figure 88 and Figure 89. In the first simulation, the Markov chain with the C transition probability defined as P(C) = 0.5 is employed[37].

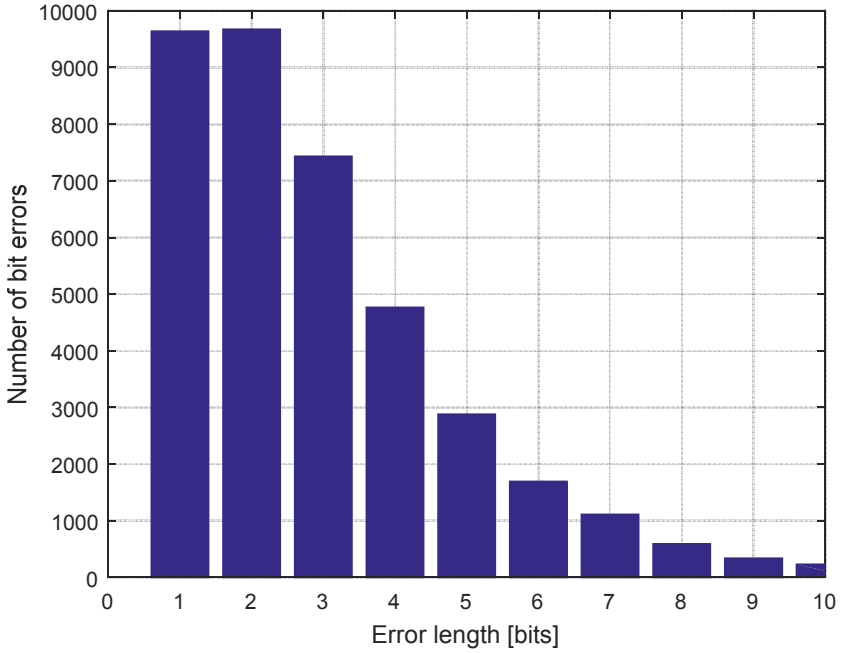


Figure 87: Error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.5

Figure 87 shows the generated error characteristic. The chain generates single and double errors mostly (50%), ~19% of triple errors, and ~12% of quadruple errors. Longer burst errors are produced infrequently. Such error characteristic corresponds to partially distorted PSSS symbols. In such conditions, RS(255,223) decoding performs better than HD-LDPC(64800,57600) and BCH(2047,1816) (Figure 88). To

[36] The PSSS processing and PSSS symbol are explained in section 1.1.8.
[37] The employed Markov chain and the chain parameters are explained is section 3.5.

compare the performance between the codes more precisely, an additional simulation with higher input BER resolution is performed, and the results are presented in a linear scale (Figure 89).
The convolutional codes cannot be considered for such conditions due to very poor error correction performance.
For BER < 2e-5 FEC coding has to be reduced or disabled. As mentioned before, link adaptation algorithm explained in section 3.12 is proposed to improve the goodput in such case.

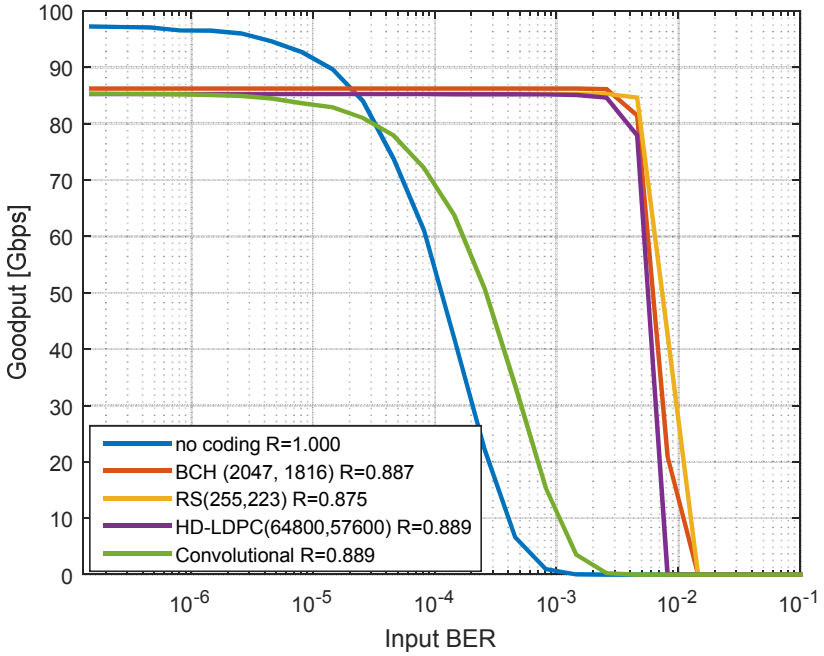


Figure 88: Comparison of FEC algorithms against theoretical error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.50 (mixed errors). The letter 'R' in the legend denotes code rate of the codes.

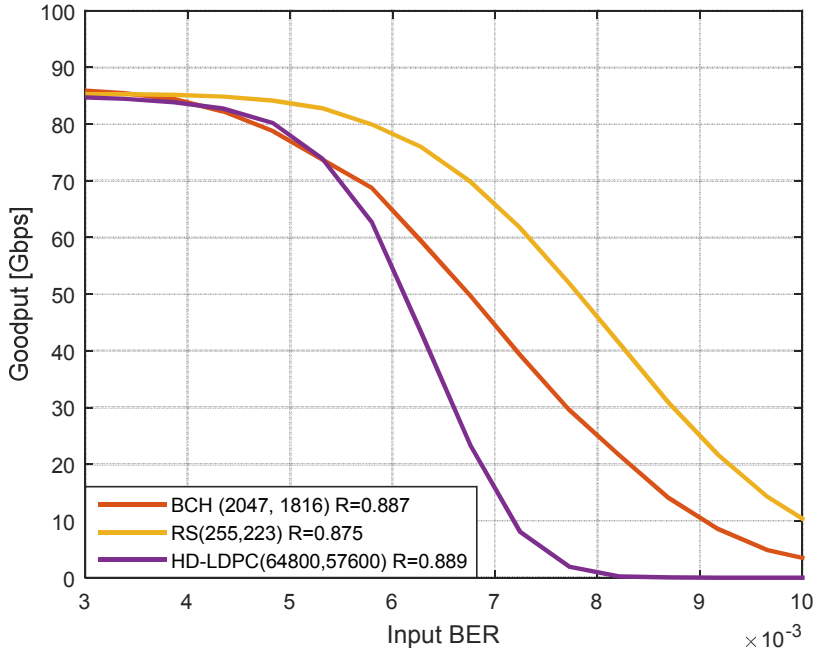


Figure 89: Comparison of FEC algorithms against theoretical error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.50. The letter 'R' in the legend denotes code rate of the codes.

3.13.3 Burst errors

If the C transition probability of the Markov chain is defined as P(C)=0.9, burst errors are generated more often than in the previous simulation (Figure 90). This error characteristics reflects a loss of a complete PSSS symbol due to noise or synchronization issues (the weighted arithmetic mean of the generated error length is equal to ~18 bits; the considered PSSS symbol size is 15 bits long). In such case, RS(255,223) decoder achieves the best results from all tested algorithms (Figure 91). BCH(2047,1816) and HD-LDPC(64800,57600) cannot outperform the RS, and an interleaver is required for the HD-LDPC to improve the decoding results. Performance of the HD-LDPC and convolutional decoders with and without interleaving is compared in sections 3.14 and 3.16.5.

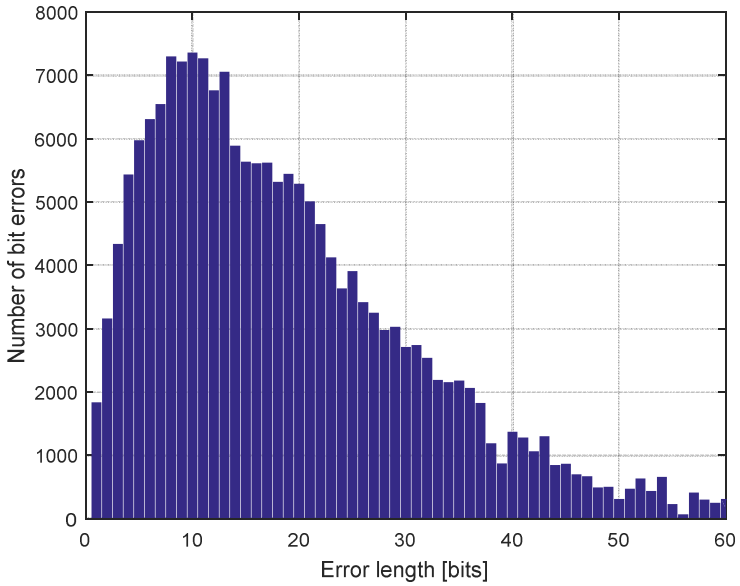


Figure 90: Error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.90.

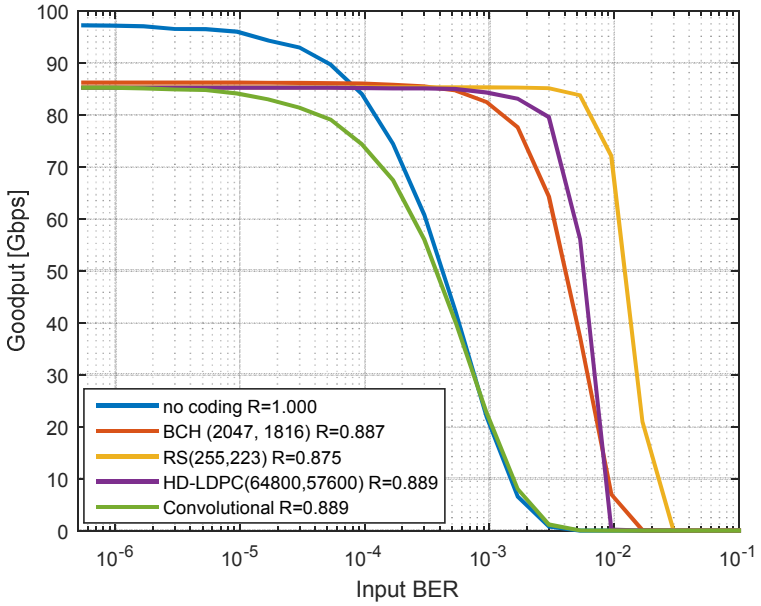


Figure 91: Comparison of FEC algorithms against for error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.90 (burst errors). The letter 'R' in the legend denotes code rate of the codes.

3.14 Interleaving

LDPC and convolutional codes achieve higher error correction performance for burst errors when the decoders use external interleaving circuits (e.g., like in DVB-S2 satellite television standard [60]). Thus in this section, convolutional codes with R = 8/9, HD-LDPC(64800, 57600), and BCH(2047,1816) decoders are tested for burst errors with interleaving[38]. Moreover, two common interleaving architectures are investigated and the minimal size of the interleavers for PSSS-15 spreading is investigated.

3.14.1 Convolutional interleaving

The general concept of convolutional interleaving is presented in Figure 92. Convolutional interleavers have two parameters that define interleaving performance of the structure shown in Figure 92. It is possible to change the number of rows and the number of memory elements in each row. The interleaver presented in Figure 92 has three rows with slope equal to two.

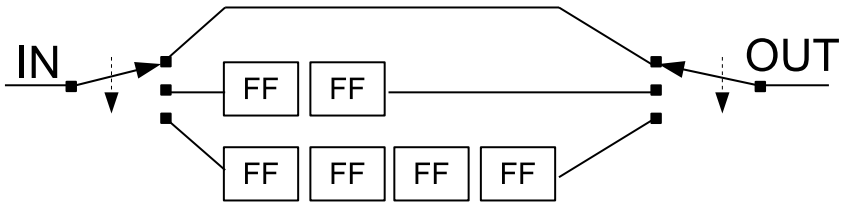


Figure 92: Example of a convolutional interleaver with three rows and slope equal to two. Figure adapted by author from [83].

The slope defines the increase of the memory elements in each next row (in the considered case 0, 2, and 4 memory elements). Both parameters have to be investigated to find the optimal architecture for the targeted PSSS-15 spreading.
Figure 93 compares performance of different convolutional interleavers with the slope in range 1 to 6 for error characteristic generated by the Markov chain with P(C) = 0.9. The interleaver with the highest slope (*slope* = 6) achieves good interleaving results with the smallest number of rows. The number of rows and the slope value are correlated with the size of the interleaver. If the number of rows and the slope value are higher, more memory elements are required to implement the interleaver (Figure 94). Additionally, the size is correlated with consumed energy. In some cases, the required chip area for an interleaver can be almost as large as for a FEC decoder (e.g., in [121] an interleaver consumes 70% of chip resources required for a soft decision Viterbi decoder with 5-bit quantization). Moreover, the distance of interleaved errors is important as well. The lowest slope (*slope* = 1) achieves the best performance according to the distance between interleaved errors, because the

[38] The idea of operation of interleavers is discussed in subsection 2.4.8.

percentage of bit errors clustered around groups with distance lower than six[39] bits is the lowest (Figure 95). Thus, during convolutional interleaver design, the tradeoff between the slope and the number of rows has to be found.

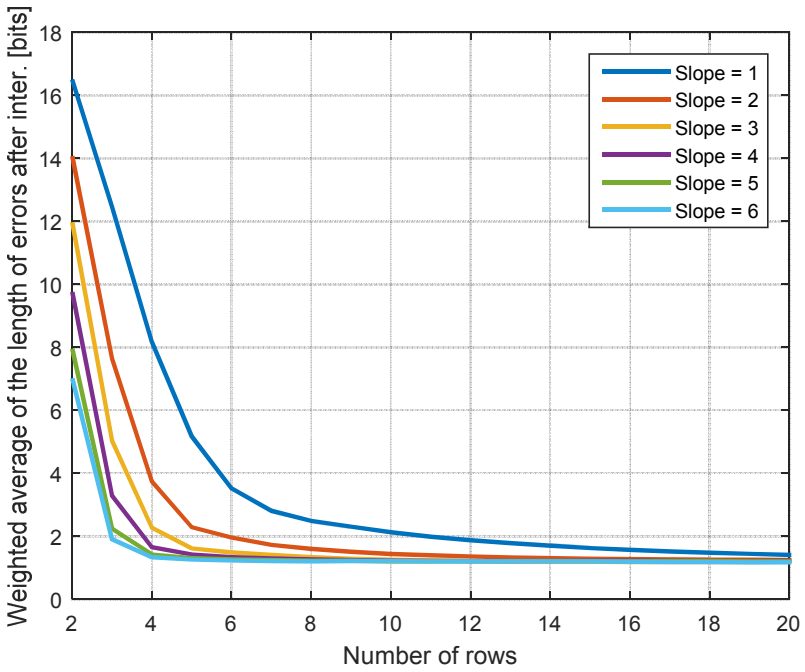


Figure 93: Interleaving performance of different convolutional interleavers with slopes in range 1 to 6. Weighted arithmetic mean of the length of errors after interleaving is taken as a comparison metric. Error characteristic is generated by the Markov chain with P(C) = 0.9.

[39] An arbitrary selected value for evaluating purposes.

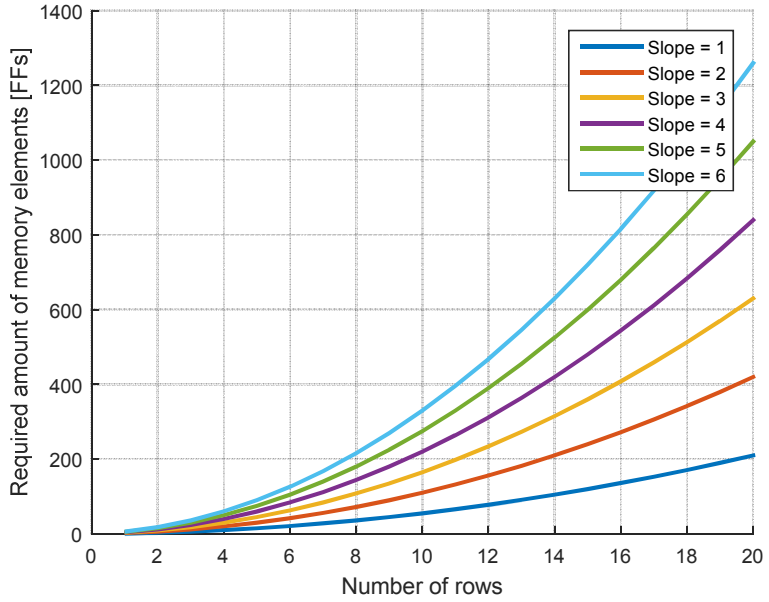


Figure 94: Required number of memory elements for interleavers with slope in range 1 to 6.

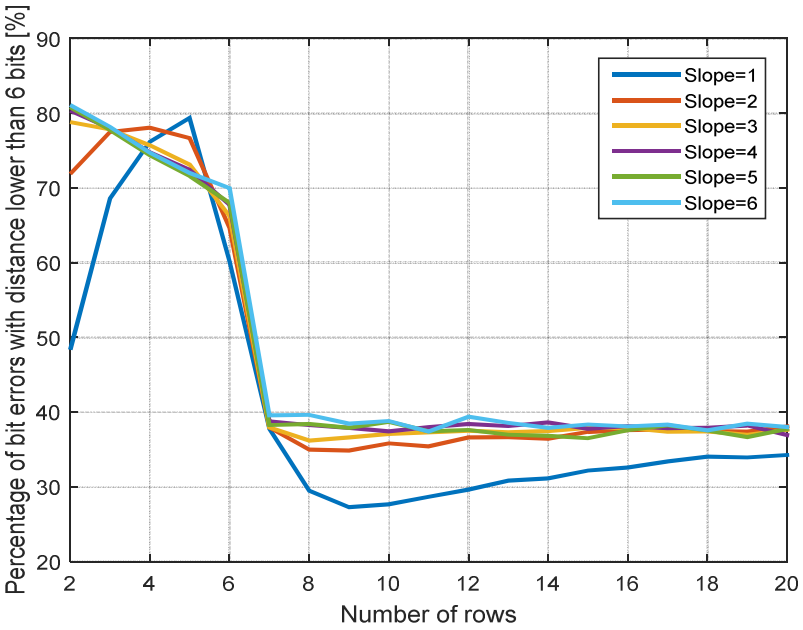


Figure 95: Convolutional interleaving - percentage of interleaved bit errors clustered around error groups with error distance shorter than six correct data bits.

3.14.2 Matrix based interleaving

The concept of matrix interleavers is presented in Figure 96. The matrix interleavers have two parameters that define the performance of bits permutation. It is possible to change the number of rows and the number of columns. The interleaver presented in Figure 96 has five rows and five columns.
In case of matrix interleavers, interleaving performance is more predictable than in case of convolutional interleavers. High number of columns reduces the average length of interleaved errors. Higher number of rows increases the distance between the errors. Figure 97 shows interleaving performance of matrix interleavers for an error characteristic generated by the Markov chain with P(C) = 0.9.
Figure 98 depicts the percentage of bit errors clustered around error groups with distance shorter than six[40] correct data bits after deinterleving.

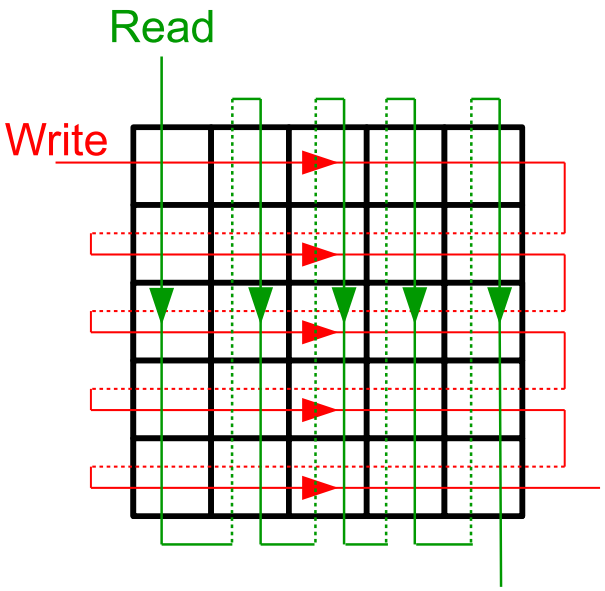


Figure 96: Example of matrix based interleaver with five columns and five rows. The arrows denoted in red show data writing order to the interleaver. The arrows denoted in green show data reading order.

[40] An arbitrary selected value for evaluating purposes.

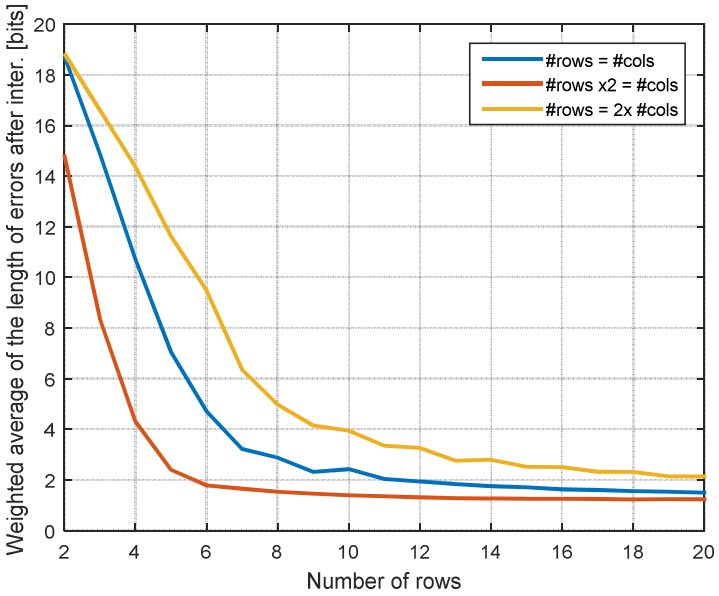


Figure 97: Performance of a matrix based interleaver. The weighted arithmetic mean of the length of the errors after interleaving is taken as a comparison metric. The interleaver is tested for error characteristic generated by the Markov chain with P(C) = 0.9.

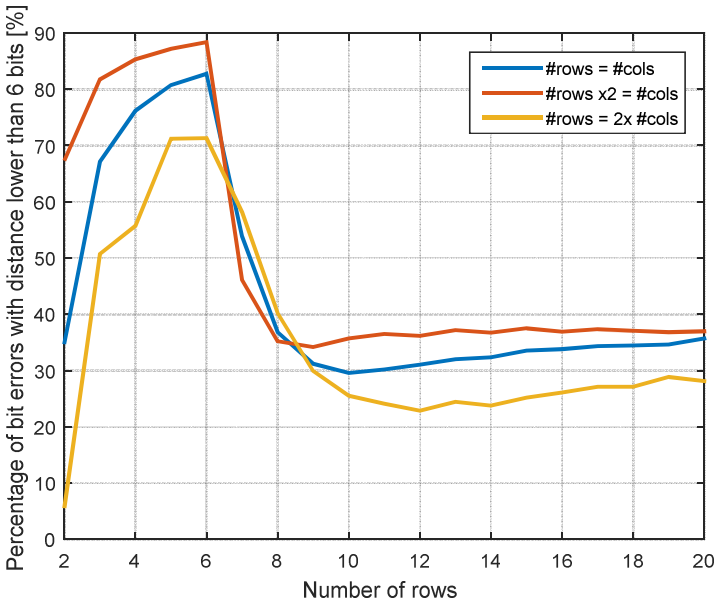


Figure 98: Matrix based interleaving - percentage of interleaved bit errors clustered around error groups, which distance between the errors is less than six correct data bits.

3.14.3 Interleavers for PSSS-15 spreading and convolutional codes

In this section, convolutional interleavers for convolutional codes are investigated. Convolutional codes compared in subsection 3.13.3 achieve very poor performance for burst errors. For the proposed error characteristic generated by the Markov chain with P(C) = 0.9, the codes do not work and do not improve the goodput in comparison to uncoded transmission. Thus, in this section, interleaver parameters for the convolutional codes are investigated. The investigated error characteristics is generated by the Markov chain with P(C) = 0.9 and are reflected in a loss of a complete PSSS symbol (the weighted arithmetic mean of the generated error length is equal to ~18 bits; the considered PSSS symbol size is 15 bits). One of recommended architectures is an interleaver with 26 rows and the slope equal to four (Figure 99). Such an interleaver consumes 1300 flip-flops (FFs) and improves the goodput from 37.18 Gbps to 80.2 Gbps. On the other hand, RS(255,239) decoder requires 213 FFs only. In case of convolutional codes, interleaver alone consumes more memory resources (FFs) than the complete RS decoder. Moreover, the RS decoder provides much higher error correction performance for burst errors than the hard decision decodable convolutional codes with interleaving.

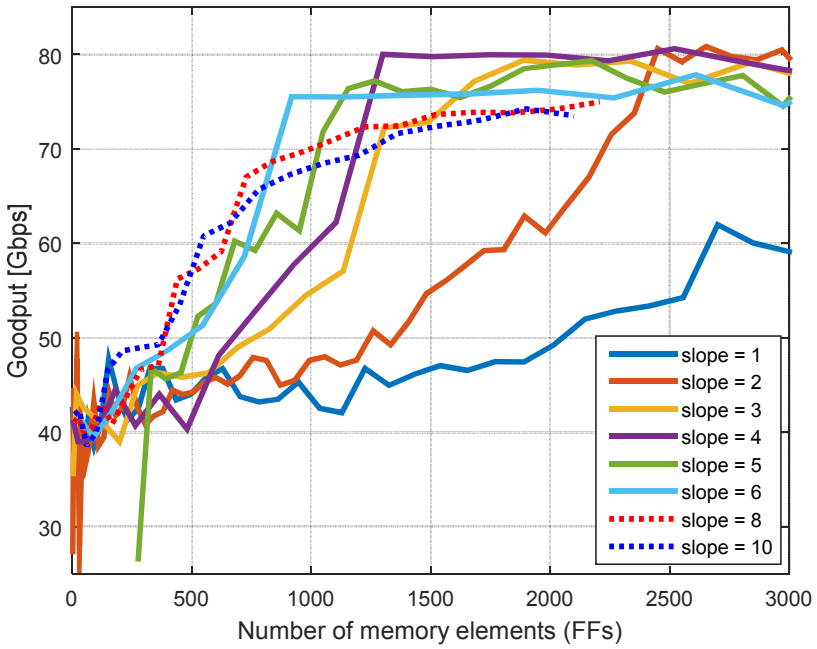


Figure 99: Results of different convolutional interleavers for convolutional codes and error characteristic generated by the Markov chain with P(C) = 0.9 and input BER = 5.33e-4. The goodput is shown as a function of the interleaver size in flip-flops (FFs).

Figure 100 proves that the proposed interleavers with slope in the range 3 to 5 are optimal for the selected error characteristic. Increasing the slope value above five

decreases the performance of the simulated system. Moreover, interleavers with size smaller than 1300 FFs are not recommended. Thus, RS codes obtain much higher error correction performance with lower hardware overhead than the presented hard-decision decodable convolutional codes with convolutional interleavers.

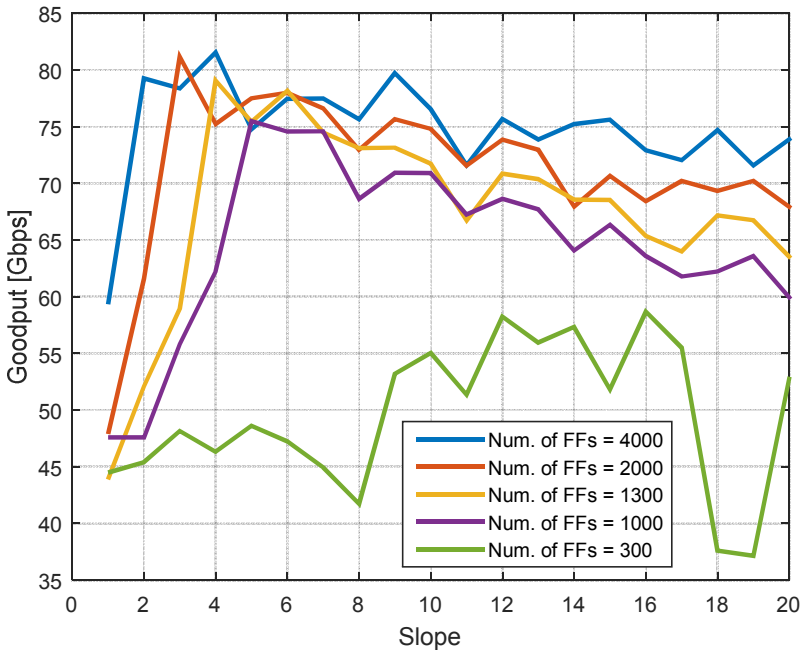


Figure 100: Results provided by different convolutional interleavers for convolutional codes and error characteristic generated by the Markov chain with probability P(C) = 0.9 and BER = 5.33e-4. The goodput is shown as a function of the slope.

Similar investigation of matrix interleavers is shown in Figure 101. For the tested error characteristic, an interleaver with matrix size of 53×107 bits (~0.7 kB) obtains goodput up to ~80.5 Gbps. A matrix of rectangular shape with doubled number of columns achieves good performance for the considered convolutional codes.
Figure 102 depicts performance of the system with the selected convolutional and matrix interleavers (*slope* = 4, *number_of_rows* = 26, and the matrix of 53×107 bits respectively). Interleaving significantly improves error correction performance of convolutional codes. However, for the presented error characteristic, relatively large interleavers are required. Thus, RS codes are recommended instead of the hard-decision decodable convolutional codes.

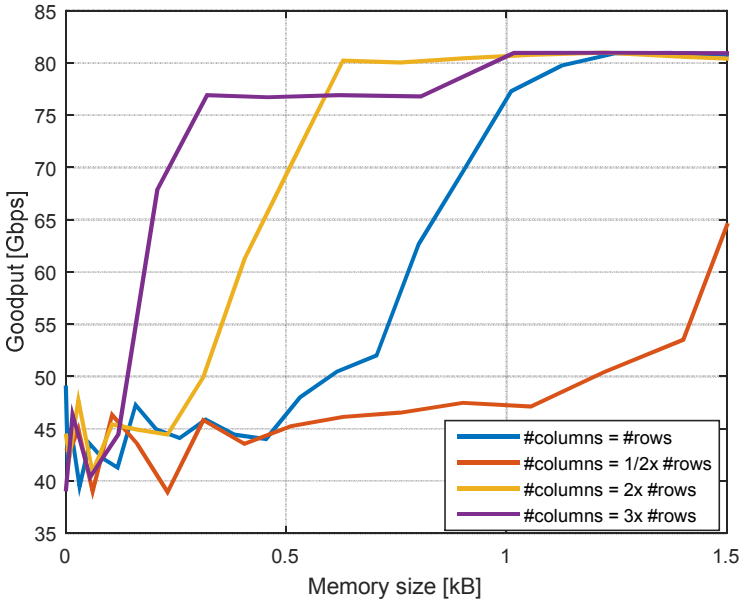


Figure 101: Results of different matrix interleavers for convolutional codes and error characteristic generated by the Markov chain with P(C) = 0.9, BER = 5.33e-4.

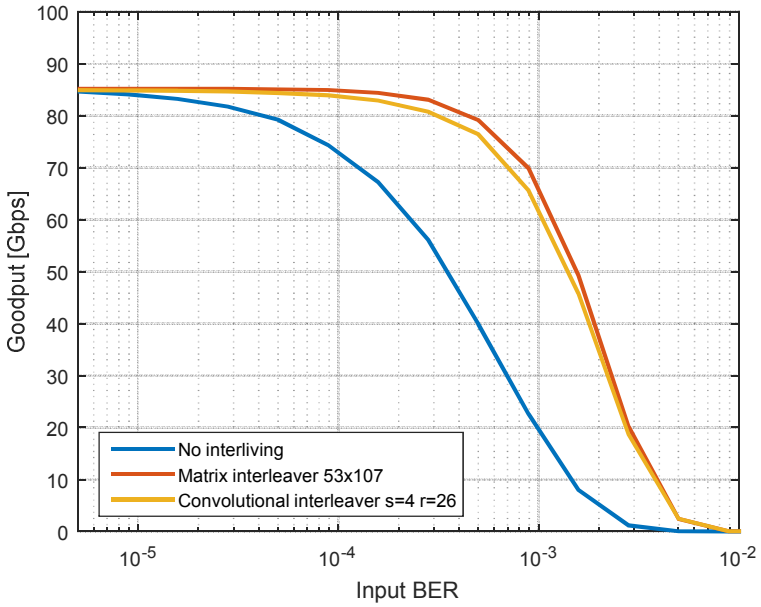


Figure 102: Results of the designed matrix and convolutional interleavers for convolutional codes and error characteristic generated by the Markov chain with P(C) = 0.9, BER = 5.33e-4.

Figure 103 depicts error characteristics generated by the Markov chain with P(C) = 0.9. Results of interleaving obtained by the selected convolutional and matrix interleavers are shown in Figure 104 and Figure 105 respectively. Both interleavers obtain similar error interleaving performance. Most of the burst errors are chopped to single errors, and the single errors represent ~76% of all errors after deinterleaving.

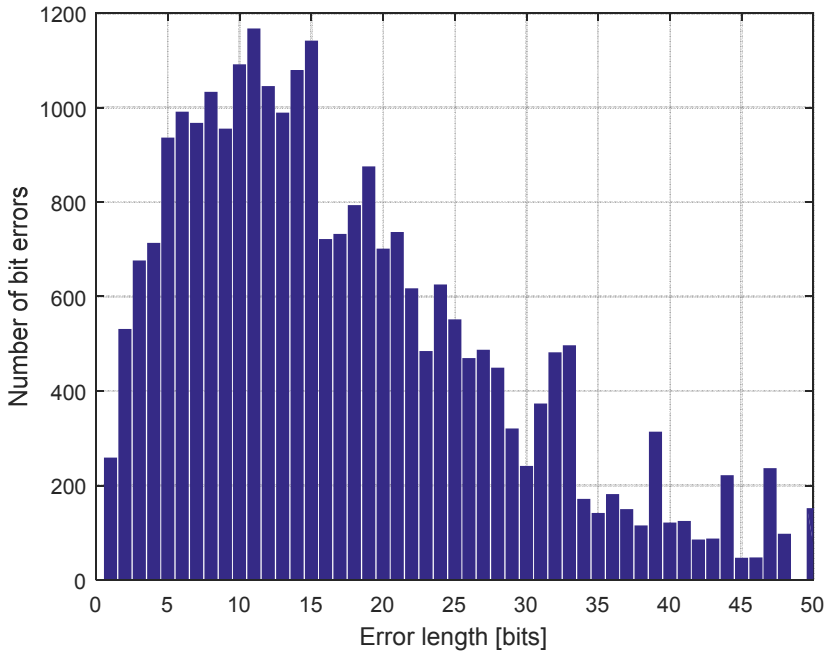


Figure 103: Error characteristic generated by the Markov chain with probability P(C) = 0.9 and BER = 5.33e-4.

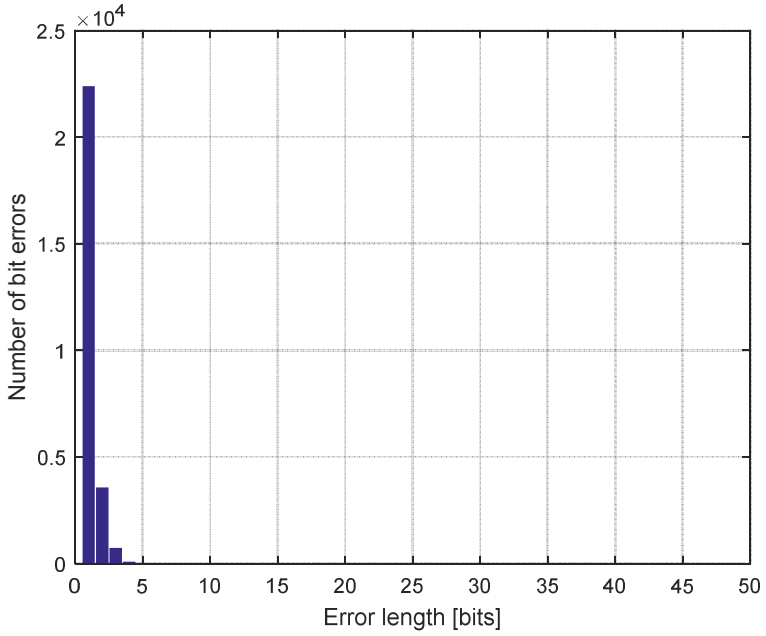


Figure 104: Error characteristic generated by the Markov chain with probability P(C) = 0.9, BER = 5.33e-4, and with convolutional interleaving (slope = 4, number of rows = 26).

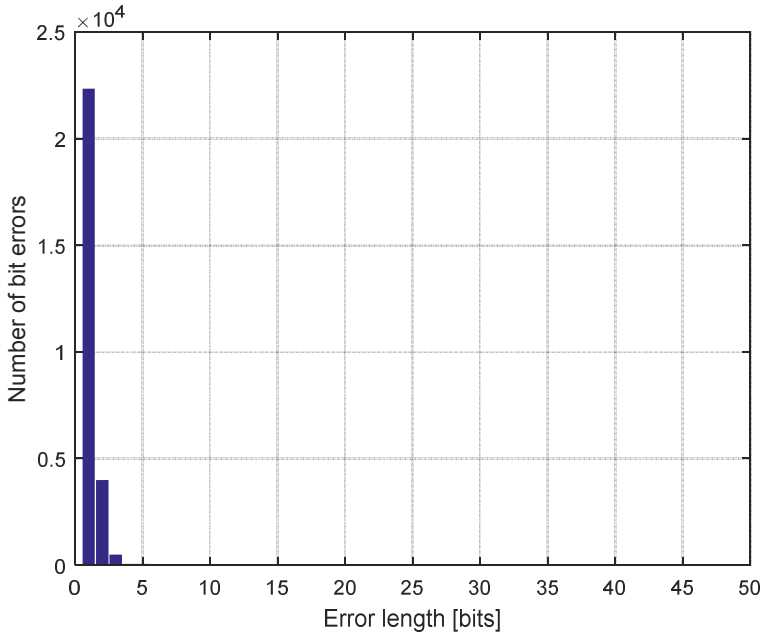


Figure 105: Error characteristic generated by the Markov chain with probability P(C) = 0.9, BER = 5.33e-4, and with matrix deinterleaving (53×107).

3.14.4 Interleavers for PSSS-15 spreading and LDPC codes

The same interleavers are simulated for the HD-LDPC decoder. In case of LDPC decoding, the differences between results obtained with and without interleaving are not so significant like in case of convolutional codes (Figure 106 and Figure 107). Moreover, the HD-LDPC code requires smaller interleavers than the convolutional code. In the presented case, a convolutional interleaver with 100 FFs, and a matrix interleaver with 100 bytes of memory provide good results. Such interleavers improve transmission goodput up to 15% at BER = ~4.5e-3 (
Figure 108). The convolutional codes tested in subsection 3.14.3 require interleavers of size of at least 1300 FFs and 700 bytes respectively.

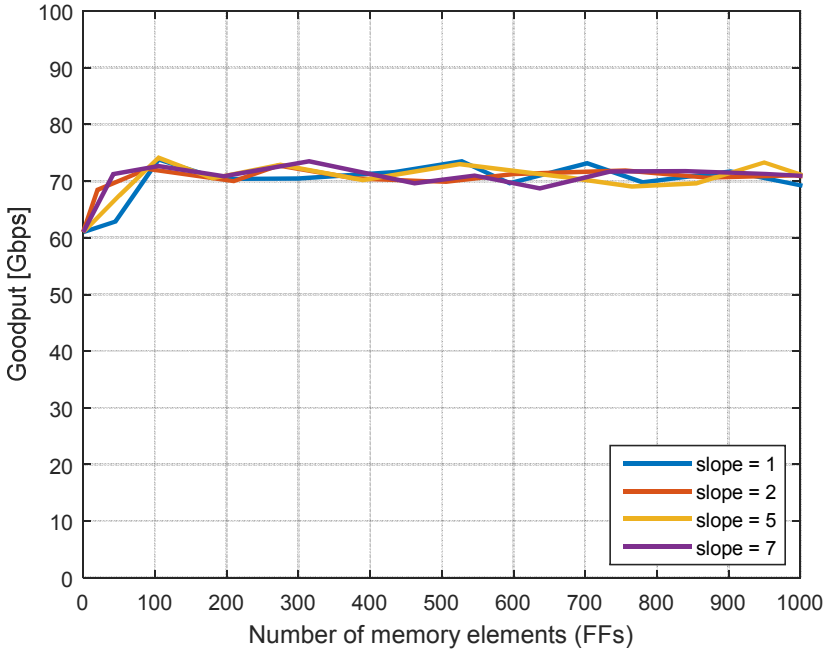


Figure 106: Results of different convolutional interleavers for LDPC codes and error characteristic generated by the Markov chain with probability P(C) = 0.9, BER = 5e-3. The goodput is show as a function of the interleaver size in flip-flops (FFs).

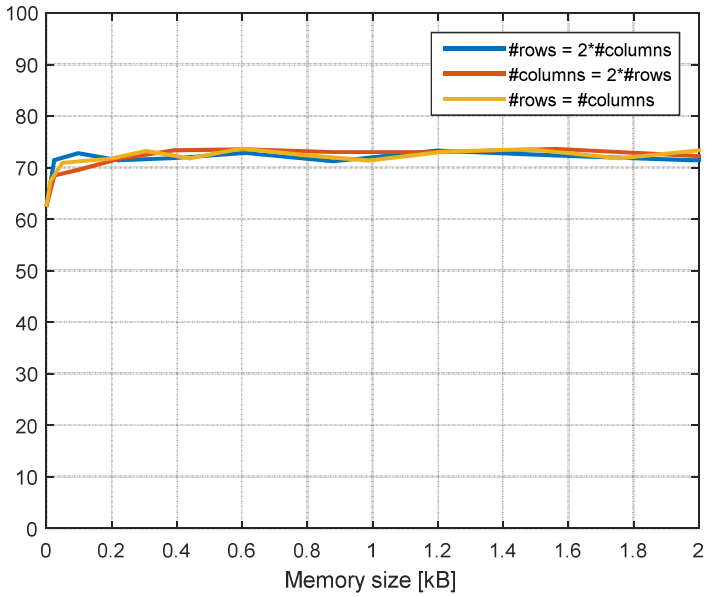


Figure 107: Results obtained by matrix interleavers for LDPC codes and error characteristic generated by the Markov chain with probability P(C) = 0.9, BER = 5e-3. The goodput is show as a function of the interleaving matrix size.

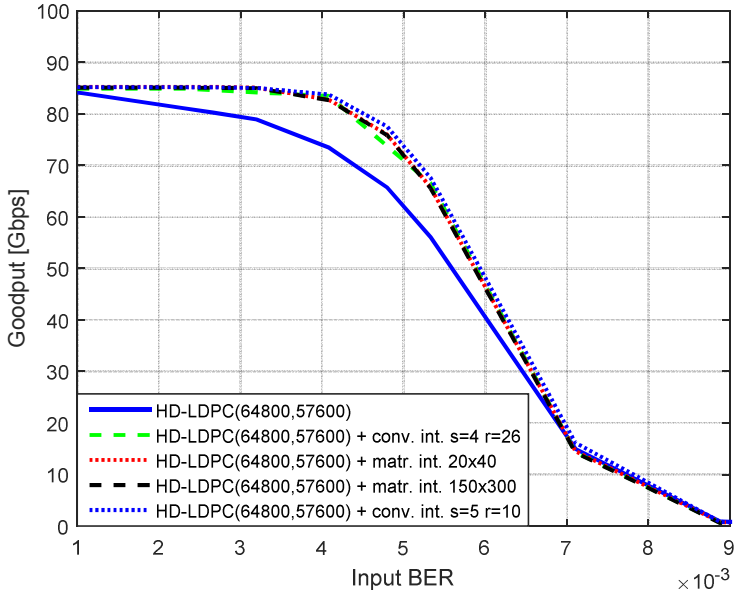


Figure 108: Comparison of selected interleavers for LDPC codes and error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.90 (burst errors).

Results of interleaving obtained by the selected convolutional and matrix interleavers are shown in Figure 109 and Figure 110 respectively. Both interleavers obtain similar interleaving performance. The results are almost identical to the results shown in Figure 104 and Figure 105, where interleaving results for convolutional codes are shown. Thus, the figures do not show the differences between the interleavers for convolutional and LDPC codes. The length of interleaved errors is almost the same, but the distance between the errors is different (Figure 111). Convolutional codes require longer distances between the errors, and the distance has to be longer than the constraint length of convolutional codes[41].

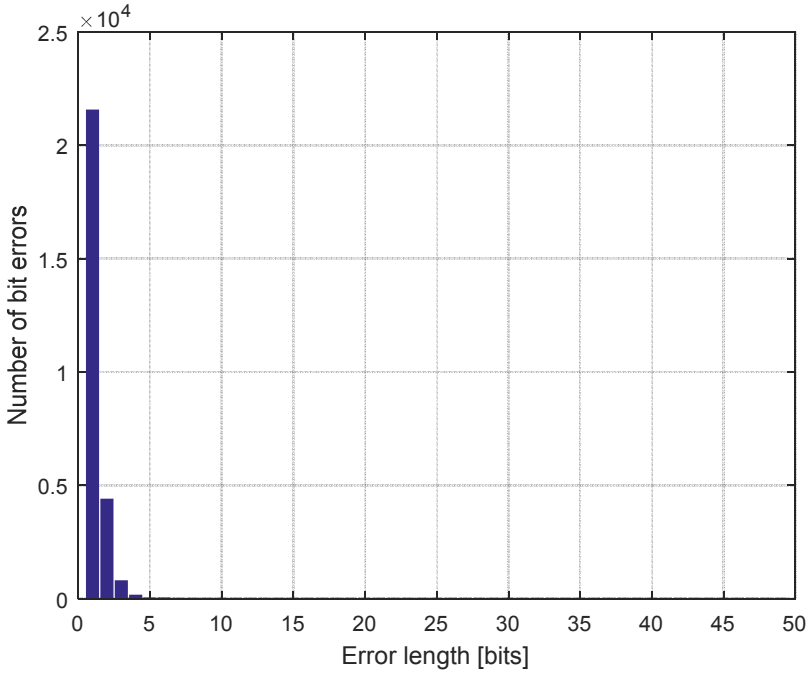


Figure 109: Error characteristic generated by the Markov chain with probability P(C) = 0.9, BER = 5.33e-4, and with convolutional deinterleaving (slope = 5, number of rows = 10).

[41] Constraint length of convolutional codes is explained in sections 2.4 and 2.4.1.

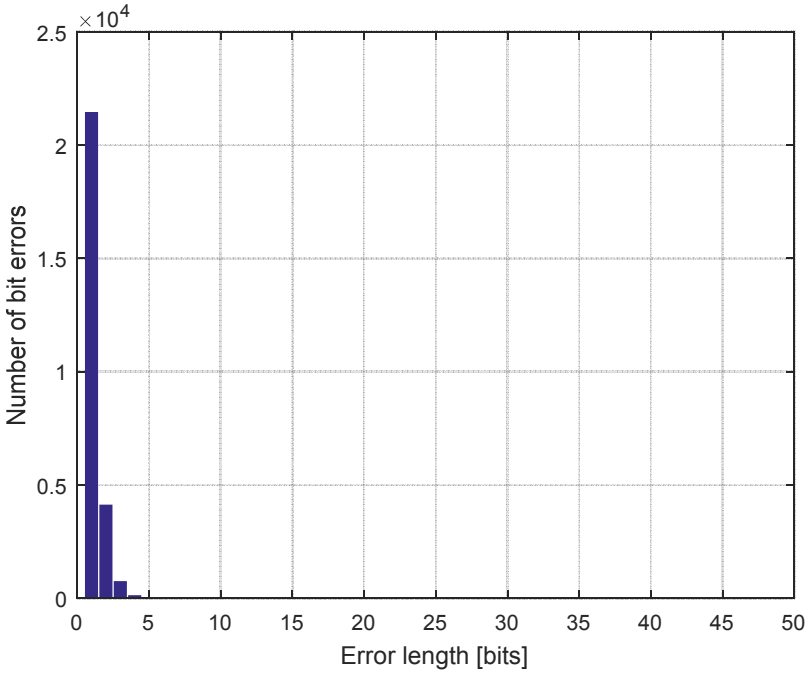
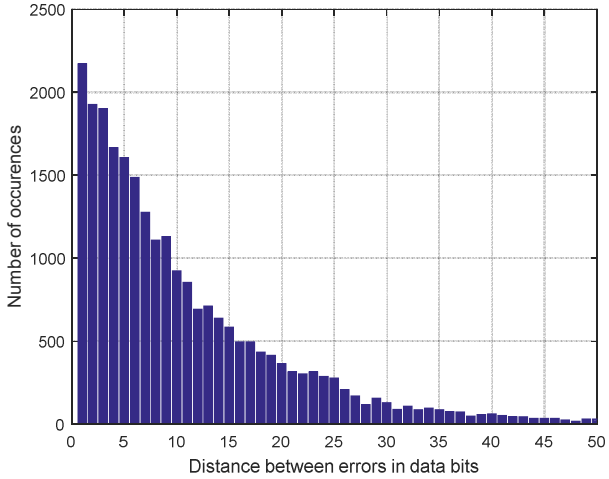
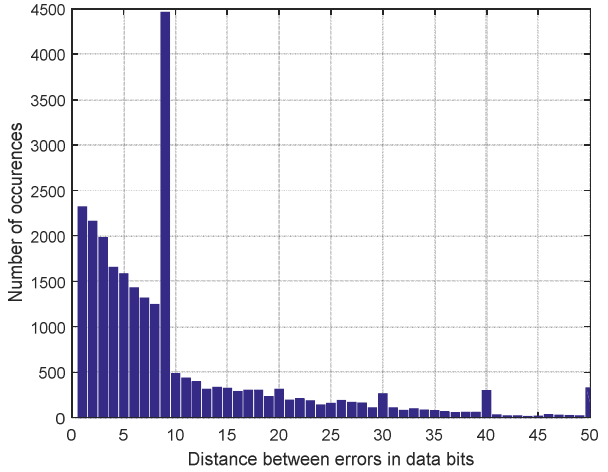


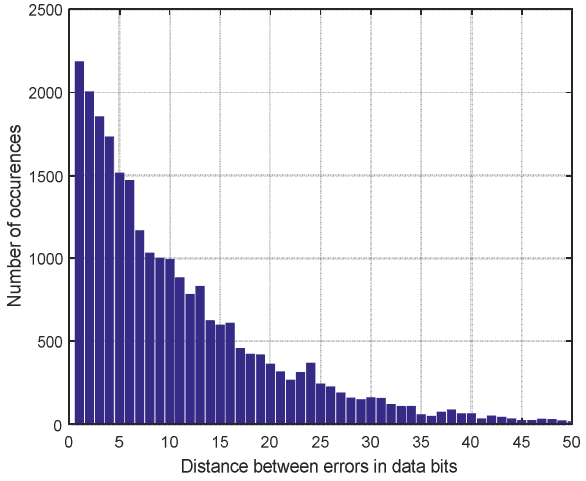
Figure 110: Error characteristic generated by the Markov chain with probability P(C) = 0.9, BER = 5.33e-4, and with matrix deinterleaving (20×40).



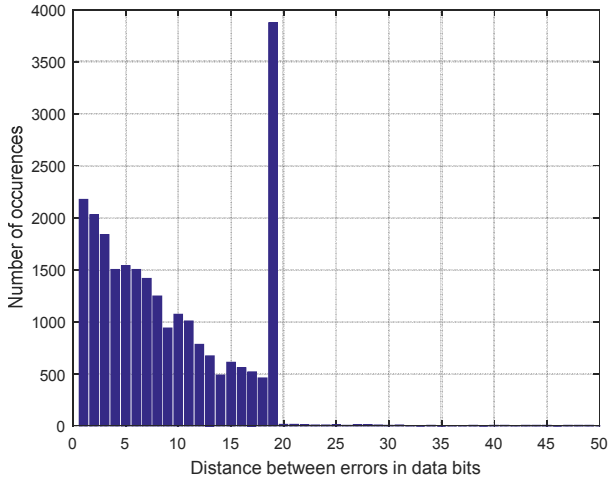
a) The convolutional interleaver with slope = 4 and number of rows = 26 dedicated for the convolutional codes.



b) The convolutional interleaver with slope = 5 and number of rows = 10 dedicated for the LDPC code. The peak at distance = 9 corresponds to the number of rows.



c) The matrix interleaver (53×107) dedicated for the convolutional codes.



d) The matrix interleaver (20×40) dedicated for the LDPC code. The peak at distance = 19 corresponds to the number of rows.

Figure 111: Comparison of interleaved error distances between the selected convolutional and matrix interleavers.

3.14.5 Interleavers for RS and BCH codes

RS codes are suitable as multiple-burst bit-error correcting codes [122], and therefore have to be used with symbol-interleavers instead of bit-interleavers. BCH codes are classified as random error correction codes [46], so an interleaver does not change the correction results. Error correction results of RS and BCH codes can be improved by an interleaver only if the interleaver spreads bit errors uniformly over neighboring code words, and the average number of bit errors per block is reduced after interleaving. Interleaving the errors within a BCH and RS code word does not improve the decoding results. For BCH and RS codes, a dedicated block-interleaving-FEC concept is discussed in section 3.15.
Figure 112 compares the performance achieved by a BCH code with and without an interleaver. As mentioned before, the interleaver used with a BCH decoder does not improve the error correction performance.

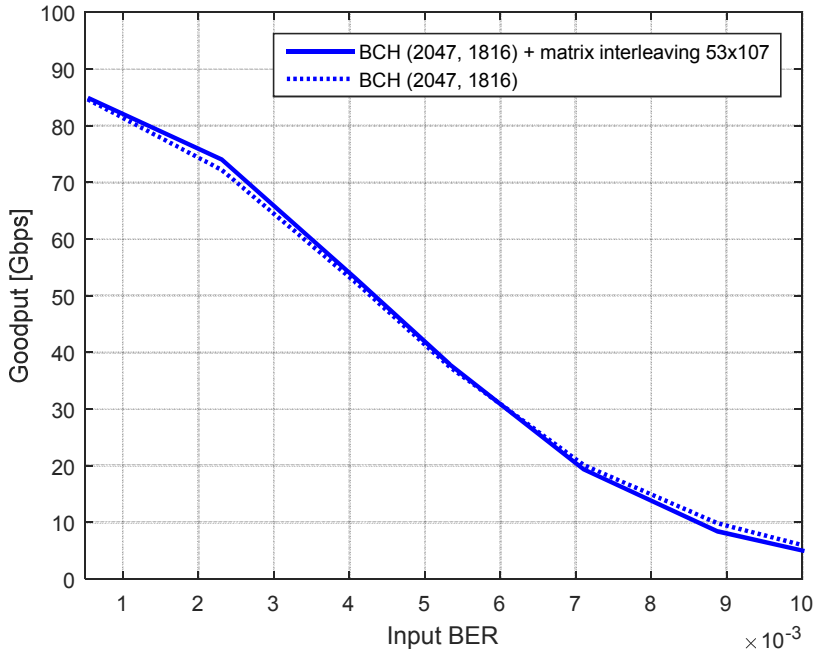


Figure 112: Comparison of BCH decoding with and without interleaving against a theoretical channel generated by the Markov chain with the C transition probability defined as P(C)=0.90. In such case, interleaving bit errors within BCH code words do not improve decoding results.

3.15 Interleaved Reed-Solomon codes dedicated for high-speed hardware decoding

In this section, interleaved Reed-Solomon (IRS) codes are investigated. The idea of operation of IRS codes is already explained in 2.4.6. Here, a modified version of IRS codes proposed for high-speed hardware decoding is proposed. FPGA technology is considered as the implementation platform, but the proposed architecture can be used for ASICs as well. The only difference is the required amount of RS decoders to achieve 100 Gbps. Moreover, the same concept can be used for construction of interleaved BCH codes.

3.15.1 IRS concept and hardware optimized IRS architecture

A single RS decoder entity with an 8-bit symbol cannot run at faster frequency than 250 MHz on Virtex7 FPGA. Thus, the throughput is limited to ~2 Gbps. It means that at least 50 parallel entities are required to achieve the targeted 100 Gbps data rate. The easiest solution to achieve the requested goodput is a parallel RS calculation array organized as an IRS decoder.
The general structure of the proposed IRS engine is shown in Figure 113. The data symbols are multiplexed between different RS decoders. Therefore, burst errors in the incoming data stream are interleaved between different RS decoders as well. Figure 114 shows hardware optimized architecture of proposed IRS decoders. The inputs and outputs accept 64-bit words. The employed RS coders are based on 4, 8, or 12-bit symbols. The last configuration does not support the 64-bit bus, but there are some other possibilities to deal with this problem. Anyway, the 12-bit words are not recommended due to moderate correction performance for single errors and high decoding complexity (this is explained further in this section).

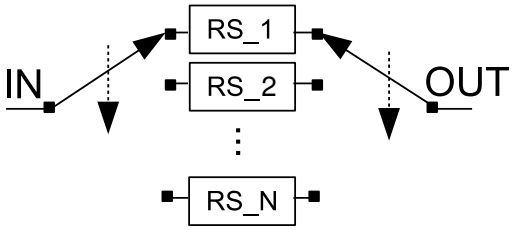


Figure 113: Interleaved Reed-Solomon (IRS) coding. Figure adapted by author from [72].

Input data is split between parallel RS structures (Figure 114). Each single RS entity calculates 4, 8, or 12 bits from the 64-bit word. The calculated amount of data is defined by the RS symbol size. The main reason of 64-bit architecture is hardware multiplexing supported by common serial protocols. The hardware multiplexers deserialize the data to 64 bits in most cases (e.g., 10G Ethernet [101] and GTH/GTX/GTZ transceivers [98]). The Ethernet, GTH, GTZ, or GTH can be used

to interface to other devices, for example to the baseband processor. Thus, 64-bit buses are considered in the design presented in this section. This significantly reduces complexity of the proposed data link layer processor.

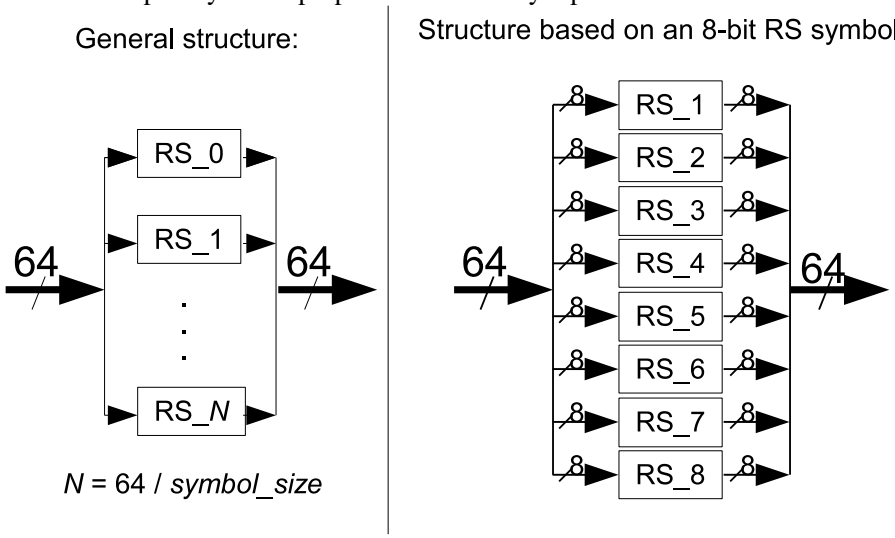


Figure 114: Proposed parallel Reed-Solomon structures for 100 Gbps IRS coding.

The proposed IRS scheme has three main advantages. Firstly, interleaving improves correction of burst errors. Secondly, the interleaver can be realized as a static routing network and there is no hardware overhead for the interleaving structure. Thirdly, IRS achieves high decoding throughput due to parallelized architecture.
There are several aspects to consider during IRS encoder design. Firstly, the optimal RS algorithm has to be chosen to build the parallel calculation array. There are many parameters to optimize e.g., symbol size, redundancy data size, error correction performance, calculation delay, logic area, and the maximal clock frequency.

3.15.2 Selection of the optimal RS algorithm

The following coding schemes are considered as a base for the IRS processor: RS(15,13), RS(255,237), and RS(4095,4006) with symbol sizes: 4, 8, and 12 bits. The code rates are on a similar level, and the decoders achieve optimal calculation latency in the targeted VHDL implementation [64], [123]. It means, the proposed codes are selected very carefully according to practical issues (the decoding throughput has to not be lower than 1-symbol / 1-clk). This assumption reduces the required clock frequency, and the area required to implement the IRS engine. Similar boundaries can be defined for other implementations of RS, and all presented observations in this section are universal for most IP-cores available on the market [61], [71], [124].
Initially, RS(255, 237, 8-bit) is selected as the default FEC-configuration for the wireless demonstrator. The 8-bit symbol size is a standard choice for most

applications, because one RS symbol is equal in length to one data byte. This simplifies data handling in software and hardware implementations. The selected RS(255, 237) requires seven times less FPGA logic than the IHP RS(255,223). Thus, Xilinx implementation of RS(255, 237) is assumed as one of targeted decoders for the IRS array.

3.15.3 Comparison of error correction performance

Figure 115 presents error correction results of the predefined hardware structures. In the figure, different symbol and code word sizes are compared. The performance depends mainly on the RS block-length but not on the symbol size. This situation can be explained in the following way. If the size of a symbol is increased, more bit errors can be corrected. However, increasing the size of a symbol increases the block length (in bits) as well. The longer block is vulnerable to bit errors and requires higher error correction performance. Thus, the extended symbol size and the extended block length are canceling each other's effects. If the error characteristic is changed, and mostly single errors occur in the communication channel, then the advantage of the longer symbol is lost. This is demonstrated in Figure 116. In this case, the shorter symbol is more efficient than the longer symbol.

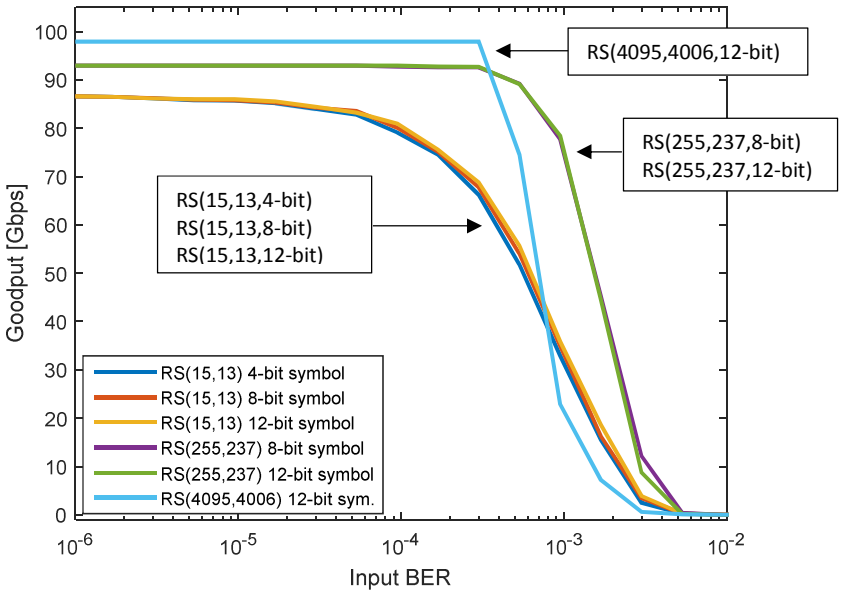


Figure 115: Comparison of error correction results for different RS codes. Transmission over a channel with burst errors is considered. In the tested conditions (burst errors), the performance depends mainly on the RS block-length but not on the symbol size.

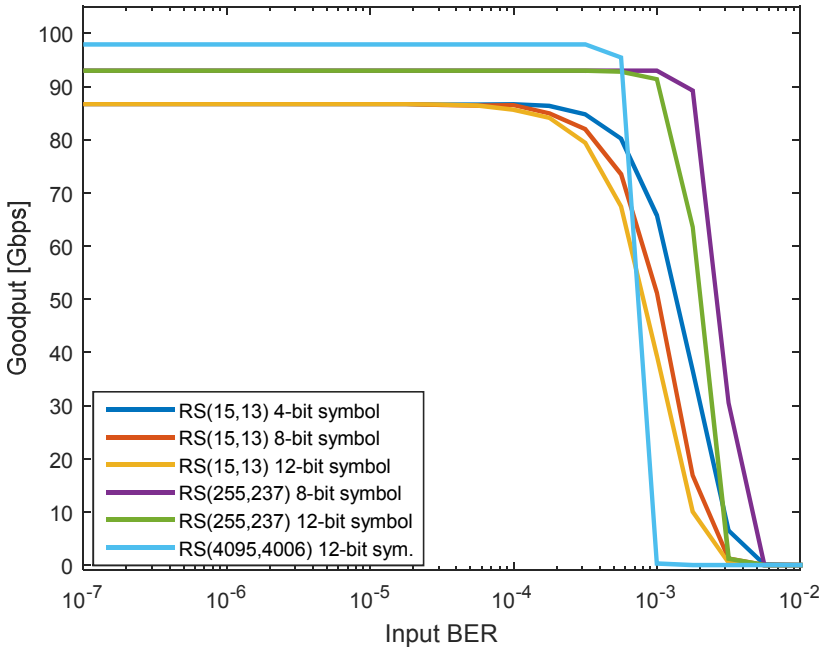


Figure 116: Comparison of error correction results of different RS codes for single errors. RS codes with smaller symbols size performs better in case of a communication channel that introduces single errors.

Furthermore, the longer block/code word (in symbols) can concentrate the redundancy data to fix the defected part of the frame. If a shorter code word is used, then the redundancy splits uniformly over a frame. In such a situation, some redundancy information is wasted in the blocks, where no bit errors occurred. This information cannot be moved to the blocks where multiple symbols are defected, and the decoder cannot correct some errors due to the limited redundancy. Therefore, the long block codes use the redundancy symbols more flexibly during the recovery process (according to the Shannon's law [125]). The symbol and block size observations lead to an important conclusion. The FEC engine should be based on a short symbol size (e.g., 4-bits) and extended code word length (255 symbols ore more). Unfortunately, the symbol size and code word length for RS codes are strongly correlated. To use a long code word (255 symbols or more) a symbol with at least 8-bits has to be employed. Therefore, BCH codes can be considered instead of RS (interleaved BCH, or TPC based on BCH, see section 3.16).
In the considered case, the IRS entities decode data in parallel to increase the decoding throughput. Such a configuration has some impact on the correction performance. Figure 117 compares the parallel decoding with a single serial decoder for burst errors. In case of the IRS processing, the errors are processed by multiple decoders. Thus, the effective length of the error is shortened, and the number of

defective symbols per decoder is reduced. This improves the correction performance due to interleaving effects of the employed hardware structure.

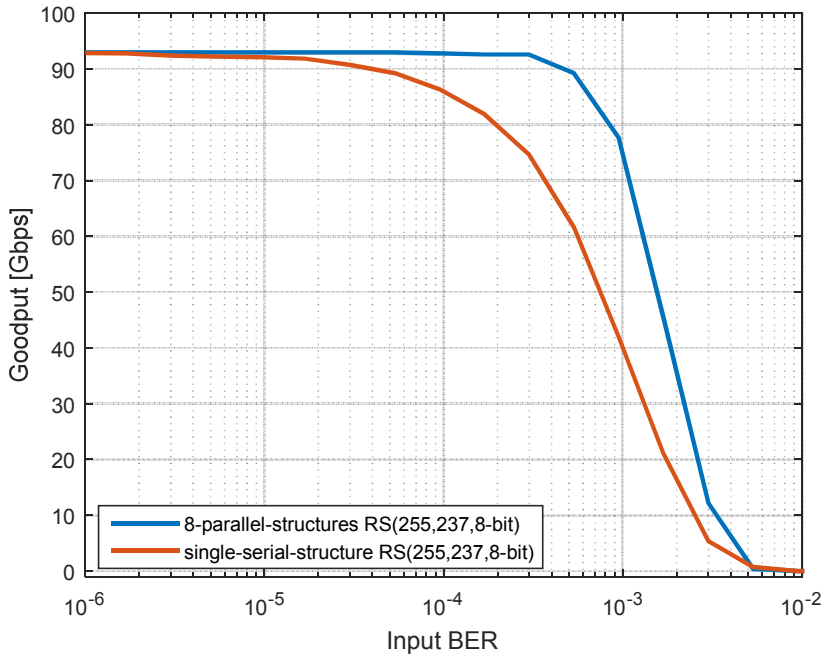


Figure 117: Comparison of burst error correction performance obtained by a single RS decoder and an array of Interleaved-RS decoders. In case of IRS decoding, long sequences of bit errors are interleaved among multiple decoders, and therefore the effective number of erroneous symbols per a decoder is reduced.

3.15.4 Hardware resources

The next considered aspect is the area consumed by each solution. Figure 118 compares resources consumed by the hardware implementation of the investigated RS decoders for Virtex7 FPGA.

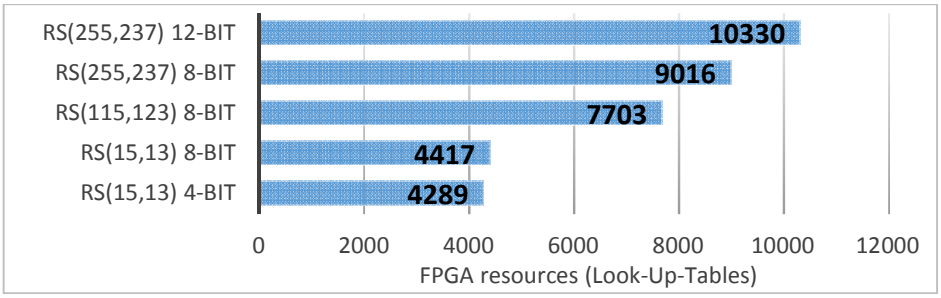


Figure 118: Comparison of consumed FPGA resources for the proposed RS decoding structures.

There were some problems with Xilinx XST synthesize tool during estimation of the results. The tool did not produce bit files for RS(4095, 4009, 12-bit), but the design was approximately two times larger than the design with RS(255, 237, 8-bit). It is demonstrated later, that the consumed FPGA area and error correction performance are correlated. However, the benefits from the minimized logic area of the reduced symbol and code word length are slightly higher than the degradation of the error correction performance.
The next considered aspect is the decoding latency generated by each solution. In the tested implementation, processing latency depends on the block length (code word) and on the size of the redundancy data. If the block length is increased, the decoder requires more cycles to decode the code word. The decoding latency does not depend on the symbol size (Figure 119).

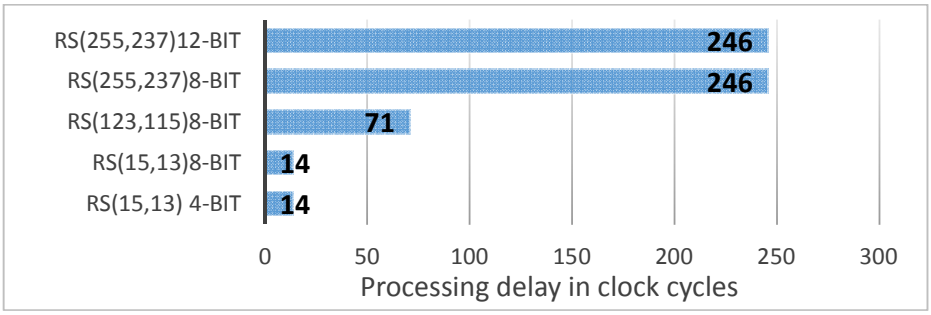


Figure 119: Comparison of RS decoding latency. RS(4095,4009,12-bit) requires 4088 cycles and is not shown in the figure.

Additionally, the design's clock constraints were changed and the layout was tested against the maximal clock frequency (Figure 120). The implementation includes the complete data link layer processor (the IRS encoders, IRS decoders, CRC LFSRs, framing, memory buffers, state machines, aggregation, deaggregation, fragmentation, and all glue logic). The design with RS(15, 13, 4-bit) runs up to 265 MHz, with RS(255, 237, 8-bit) up to 220 MHz, and with RS(255, 237, 12-bit) up to 210 MHz.

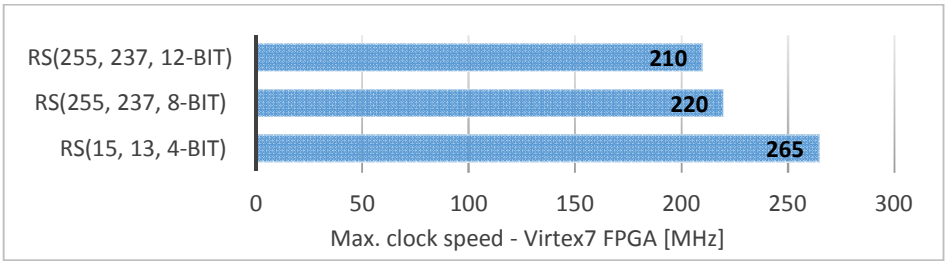


Figure 120: Maximal clock frequency for the proposed IRS designs.

3.15.5 IRS summary

Obviously, the selection of the best RS configuration is not trivial and depends on several factors. It is expected that the error correction performance is strongly correlated to the generated redundancy data size. However,
Figure 121 shows that the distribution of the redundancy data is important as well.

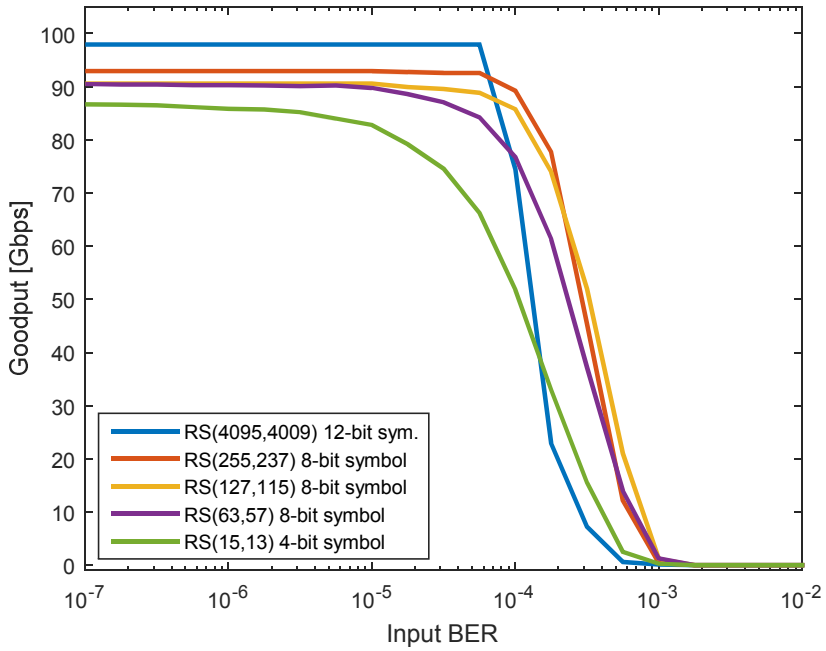


Figure 121: Comparison of error correction performance for different IRS codes.

RS(4095, 4009, 12-bit) is more efficient at channels with lower BER, but due to limited redundancy cannot correct all errors at the channels with higher BER. Additionally, required hardware area and processing delay is not acceptable for the selected application. Switching to 4-bit symbol reduces the implementation size and the decoding latency, but error correction performance of RS(15, 13, 4-bit) disappoints. This solution cannot be considered for the final wireless demonstrator as well. It means, only RS(127, 115, 8-bit) can be selected instead of RS(255, 237, 8-bit). Firstly, the decoding latency can be reduced by 71% (according to Figure 119). Moreover, the FEC logic area can be reduced by 14% (Figure 118) at the cost of degradation of the correction efficiency by around 3% (Figure 121). As this work focuses on achieving high data rates and not saving hardware resources, this solution is not considered for now. If the latency plays a significant role in a design, then RS(127, 115) and RS(63, 57) are more attractive options. However, when link adaptation approach is considered, the original RS(255, 237, 8-bit) can be used to generate a family of similar codes with redundancy in range of 2-18 symbols per a

single RS block. These codes can be implemented in a single coder with a controlling interface. This allows to apply the link adaptation, which selects the best code according to the channel condition. Such approach is investigated in section 3.12. The mentioned 8-bit codes can be fit into Virtex7 FPGA, and it is possible to achieve 100 Gbps goodput using a single FPGA chip. The implemented IRS solution uses 80 RS decoders and can correct a burst error with length up to 5760 bits. This corresponds to 384 PSSS symbols and to transmission time of ~58 ns at the targeted data rate.

3.16 Proposed improvements for turbo product codes

Turbo product codes[42] (TPC) use the same concept of interleaving like IRS codes. However, in case of TPC coding, the code words overlap. Figure 122 compares IRS codes with TPC codes. IRS decoding proposed in this work (section 3.15) uses horizontal code words calculated over rectangular matrix of symbols. TPC codes use column code words additionally. Thus, each data bit is protected by two independent code words, and iterative decoding is employed to improve error correction performance.
This section discusses an improved coding scheme for TPC codes. Error correction performance and energy efficiency per bit of the new method is significantly higher as compared to the state of the art TPC approach.

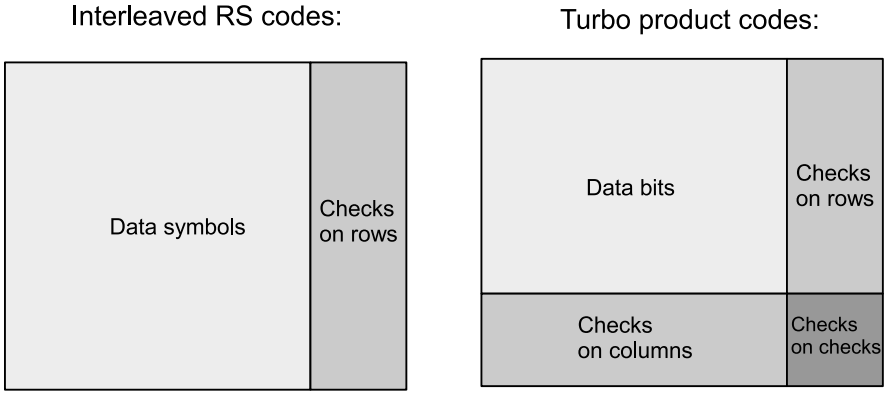


Figure 122: Comparison of interleaved Reed-Solomon (IRS) codes and turbo product codes (TPC). TPC decoding can be represented as two interleaved BCH (or RS) decoders, which decoding is performed for rows and columns iteratively. The right sight of the figure is adapted by author from [89].

[42] The idea of operation of turbo product codes is explained in subsection 2.4.12.

3.16.1 TPC in 100 Gbps optical communication systems

Forward error correction mechanisms for optical communication systems are defined in the G.709, G.975, and G.975.1 ITU recommendations [126]–[128]. Some of them evolved to more powerful decoding schemes and at least one of the algorithms can be used as a base for 100 Gbps FEC processor for wireless communication. Hard decodable turbo product codes (HD-TPC) with shortened BCH(2047,2014,t=3) component code presented in [90] achieves high performance in the class of comparable codes [89]. Additionally, the authors justify that the algorithm can run on an FPGA device with required goodput of 100 Gbps [90]. This is especially important for implementation of the DLL processor presented in this work. Furthermore, it is already demonstrated in [35] that 100 Gbps Viterbi decoder with 5-bit soft coding [66] consumes logic area of 23 Virtex7 FPGAs. Thus, the FEC code used in 100 Gbps applications has to be selected very carefully to avoid big hardware overheads. From computational complexity point of view, hard-decision decoding (HD) procedure used in the mentioned TPC approach reduces computation effort, and does not require an ADC for soft-bit quantization. Thus, hard-decision (HD) approach is preferred. All future research dedicated to the highest speed communications will probably require HD methods due to limited computation power, connectivity, and ADC technology. The simplified HD decoding can still be an interesting option in some areas, where more powerful soft-decision decoding (SD) cannot be used (e.g., prototyping of ultra-high-speed transmissions).
In this work, an improved HD-TPC decoder is presented. The newly proposed decoder improves decoding results in terms of input BER, reduces the number of required decoding iterations, and consumed energy per bit.

3.16.2 Detailed description of TPC proposed by Li et al.

The idea of operation of TPC codes is already explained in subsection 2.4.12. Here, a detailed description of the TPC codes prosed by Li et al. in [90] (denoted as Li-TPC) is introduced. The authors use shortened BCH codes to construct a TPC decoder with 16% redundancy overhead and matrix size of 390×390 bits (Figure 123). In this section, weakness of the solution are identified, and in subsection 3.16.3 an improved decoding scheme is introduced. The proposed improvements allow to correct higher input BER, require less energy per bit for decoding, and allow to achieve high decoding throughput when implemented in hardware.
First of all, the base solution uses BCH(2047,2014,t=3) component code (Figure 124). The code can correct up to $t = 3$ errors, and in case of four or more bit errors in a code word, the decoding error probability is equal to $1/(t!) = 1/(3!) \approx 16\%$. It means that there is a chance of around 16% that a code word with more than three errors will be decoded to a message different from the original one. The probability is very high, so the number of bit errors will probably increase during the iterative TPC decoding, and error correction performance will be significantly reduced. There is no possibility to detect and recover from such situation. Shortly speaking, such decoder cannot work because there is no possibility to detect and successfully decode

a row or a column with four or more bit errors. To overcome this problem, the authors shorten BCH(2047,2014,t=3) code to BCH(390,357,t=3).

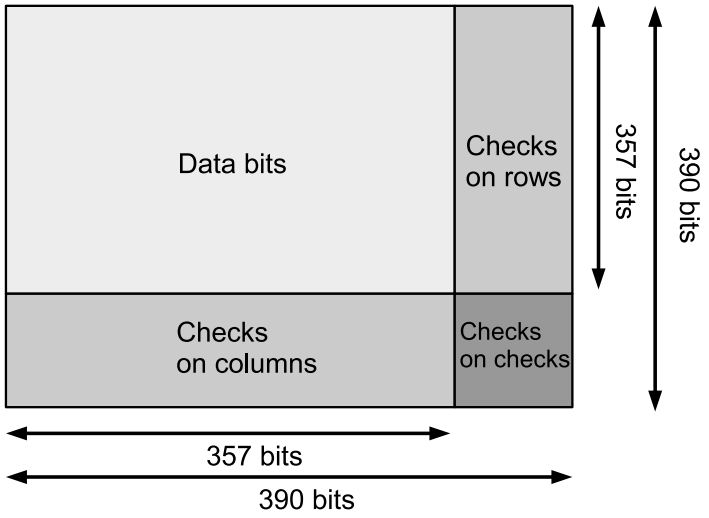


Figure 123: Turbo product code (TPC) decoding matrix proposed for optical communication by T. Li et al. in [90]. *Figure adapted by author from* [90].

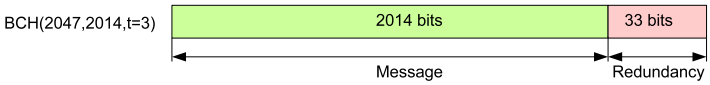


Figure 124: BCH code used in the TPC approach proposed by T. Li et al in [90].

The unused 1657 bits are set to '0' during encoding and checked after decoding if the '0'-padding pattern is unchanged (Figure 125). This improves the decoding error probability to ~0.1% [90]. This modification is the most important feature of the Li-TPC decoder, and allows to employ 3-bit correcting codes in the TPC approach. The shortened code is used for rows and columns encoding of the TPC matrix as shown in Figure 126. The shortening improves decoding results but also extends the required number of redundancy bits and decoding effort. BCH(2047,2014,t=3) code uses 33 parity bits, but in this case BCH(511,484,t=3) with 27 parity bits would be enough to correct the same number of errors. However, BCH(511,484,t=3) cannot be used due to the very high decoding error probability.

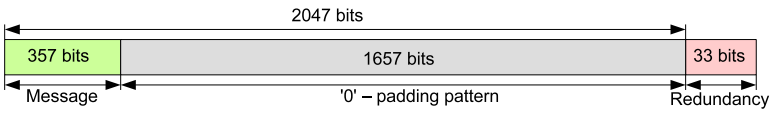


Figure 125: Shortened BCH(390,357,t=3) based on BCH(2047,2014,t=3).

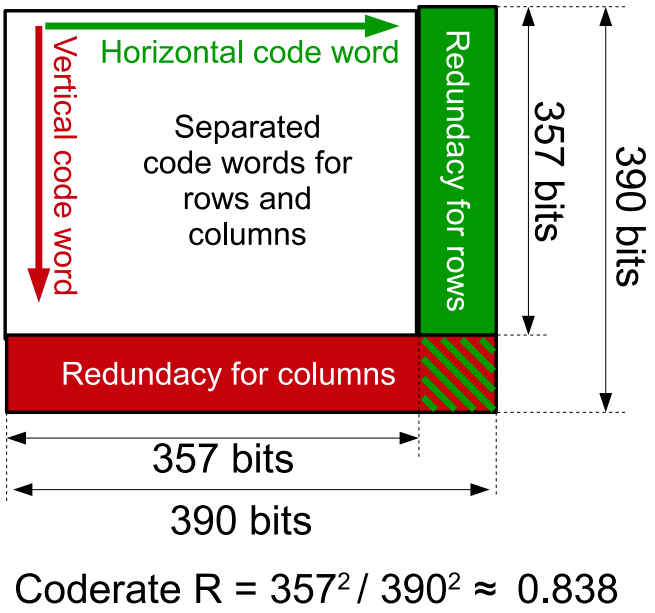


Figure 126: TPC processing matrix as proposed in [90] *(detonated as Li-TPC method).*

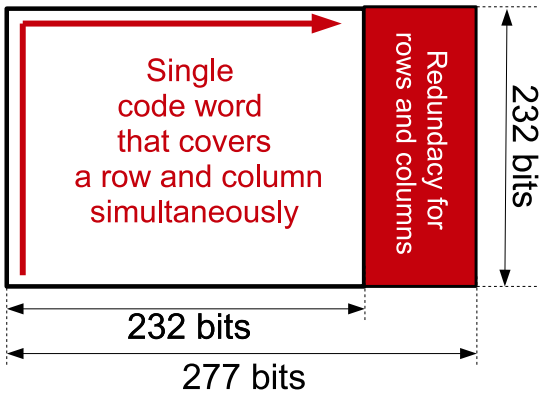
3.16.3 Proposed improvements

In this work, three improvements for the TPC scheme presented by Li et al. [90] (denoted as Li-TPC) are proposed. Firstly, shortened BCH(390,357,t=3) code is replaced by BCH(511,466,t=5). The newly proposed code with $t = 5$ has relatively low decoding error probability ($1/t! \approx 0.8\%$) and do not have to be shortened. Alternatively, BCH(1023,963,t=6) with decoding error probability defined as $1/t! \approx 0.13\%$ can be used.

The second proposed improvement is replacement of the vertical and horizontal BCH decoders with a single code that covers a column and a row simultaneously (Figure 127). The Li-TPC method presented in [90] uses two separated code words. One code word is dedicated for columns encoding, and another code word for rows encoding (Figure 126). If a single code word is used, the TPC decoder uses smaller decoding matrix. Additionally, the effective code rate of the decoder can be increased. For example, a classical TPC with an independent row and column decoder based on BCH(511,466,t=5) requires decoding matrix of 511×511 bits with code rate $R = 0.831$. However, the method proposed in this work can use matrix size of 232×277 bits with effective code rate $R = 0.837$ (matrix size is reduced four times, code rate is increased by 0.7%).

The third proposed improvement is the usage of a single code word with horizontal-diagonal shape crossing at an angle of 45º, instead of horizontal and vertical code words crossing at 90º (Figure 128). In Figure 127 the code words are crossing at an

angle of 90º, but such architecture is inefficient for burst errors correction. Thus, the decoder shown in Figure 127 is modified to the decoder shown in Figure 128.



$$\text{Coderate } R = 232^2 / (232 \times 277) \approx 0.838$$

Figure 127: Modified TPC processing matrix (proposed improvement).

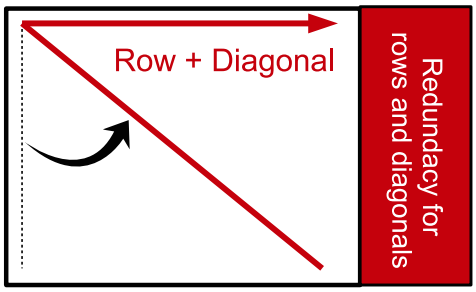


Figure 128: The newly proposed TPC code word with horizontal-diagonal shape crossing at an angle of 45º.

The proposed code word crossing at 45º achieves similar performance for single errors like the code word crossing at 90º, but the code word with 45º is more efficient for burst errors correction. This can be explained in the following way. Presuming that data writing to the decoding matrix is performed in column order (Figure 129), burst errors are aligned with the vertical code word (Figure 130). In such case, the vertical BCH decoder is blocked during the decoding. The vertical code words cannot be decoded, because the number of bit errors in the code words may exceeds the BCH correcting capacity, e.g., 3 or 5 bits in the considered decoders. If the burst error is longer than 3 or 5 bits, the vertical code word with the burst error cannot be decoded. Thus, the correction effort is moved to the horizontal code word, and error correction performance is reduced. Alternatively, if data is written in the row order to the TPC matrix, the horizontal code word is blocked by burst errors, and the

decoding effort is moved to the vertical code word. The problem is the same, one of the code words is blocked by burst errors, and only 'half' of the decoder corrects errors.

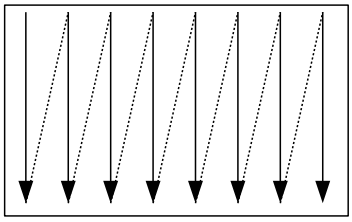


Figure 129: Data-writing order to the TPC decoding matrix recommended for Virtex7 GTX/GTH hardware.

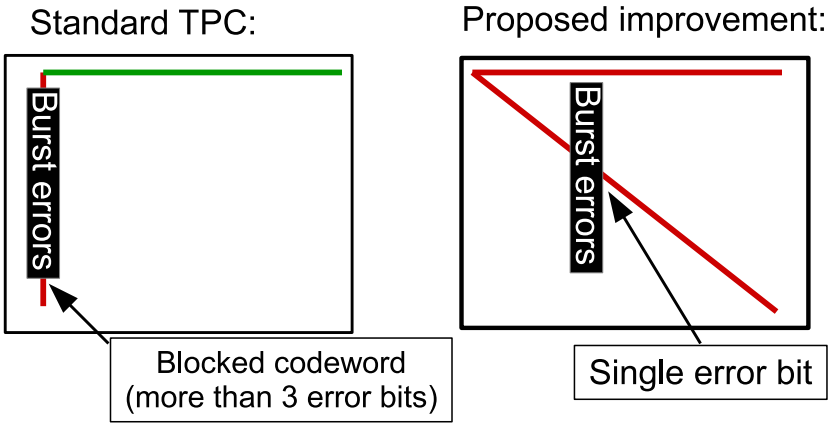


Figure 130: Blockage of the state of the art TPC decoder by burst errors (right side), and the proposed solution to overcome the problem (left side).

In the approach proposed in this work, both parts of the code word (horizontal and diagonal) are never aligned to the vertical burst errors (writing to the decoding matrix has to be performed in column order). The diagonal part of the code word interleaves burst errors and cannot be blocked like in the Li-TPC solution presented in [90]. The same idea can be used to construct an RS-based decoder (denoted as RS-TPC).

3.16.4 Analysis of error correction performance and decoding effort

The proposed improvements allow to build a TPC matrix with code rate equal to the Li-TPC method presented in [90] ($R = 357^2/390^2 \approx 0.8379$). Here, BCH(511,466,t=5) has been selected for all further investigations. Decoding of BCH codes with error correction ability t=5 or t=6 is higher than for BCH codes with t=3. On the other hand, the newly proposed code word is significantly shorter. Table 2 compares throughput measured on Intel i7-4702MQ processor for the three mentioned BCH

codes. The selected BCH(511,466,t=5) achieves 30% lower decoding throughput than BCH(2047,2014,t=3). Anyway, BCH(511,466,t=5) with 232×232 data matrix seems to be a good competitor against the Li-TPC solution and requires lower decoding effort than the alternatively proposed BCH(1023,963,t=6) code.

	BCH (2047,2014,t=3)	BCH (511,466,t=5)	BCH (1023,963,t=6)
Throughput	8.3 Mbps	5.8 Mbps	5.14 Mbps
Code rate	~0.984	~0.912	~0.941

Table 2: Throughput of BCH(2047,2014,t=3), BCH(511,466,t=5), and BCH(1023,963,t=6) codes measured on Intel i7-4702MQ processor.

Both TPC solutions are characterized by the same code rate ($R \approx 0.838$), but error correction abilities are different. The Li-TPC approach corrects up to three errors in 357 bits (3 errors in a single row or column). It means, one corrected error every 119 data bits (error correction ratio is equal to $3/357 \approx 0.84\%$). However, the newly proposed solution (denoted as BCH-TPC) corrects 5 errors in 464 bits (*column_length* + *row_length* = 464). It means, one corrected error every 92.8 data bits (error correction ratio is equal to $5/464 \approx 1.08\%$). Therefore, the new solution (denoted as BCH-TPC) performs better against bit errors and provides the same code rate as the Li-TPC method proposed for fiber optic communication [89], [90].

3.16.5 Performance of the improved TPC decoding scheme

To get a good overview of error correction performance, BCH-TPC and RS-TPC solutions are compared to HD-LDPC and the Li-TPC decoders proposed in [90]. Due to comparison reasons, the Li-TPC, BCH-TPC, and RS-TPC are modified to the code rate of the DVBT-S2 5/6 LDPC code [60]. The BCH-TPC and RS-TPC use shortened BCH(511,466,t=5) and RS(176,160) codes respectively. All four codes have exactly the same code rates (the same normalized number of redundancy bits). The BCH based decoders are run up to 10 iterations, but the RS only up to 4. This is done due to high RS decoding latencies. The RS-TPC solution may require up to 200 cycles for single iteration decoding [64]. Therefore, the RS solution has to be run with limited number of iterations. In other case, decoding of the RS-TPC matrix may introduce too long latencies for practical applications. All of the compared codes are tested against the same amount of data bits. The HD-LDPC is used to show decoding limits of the hard-decision decodable algorithms. Indeed, it is a soft-decision decoding LDPC decoder with a binary quantized input on the first decoding stage. Additionally, the decoding routine performs up to 50 iterations.
In Figure 131, all algorithms are tested against single errors. In this case, the HD-LDPC decoder provides the best results and achieves slightly higher error correction performance than the newly proposed BCH-TPC decoder. Up to BER = 1.6e-2 both algorithms achieve almost the same error correction performance, but above the value, HD-LDPC performs better. The standard TPC (Li-TPC) method is less effective than the new BCH-TPC method. The new decoder corrects up to 24%

more bit errors. The RS-TPC provides poor results and cannot be considered for this channel type.

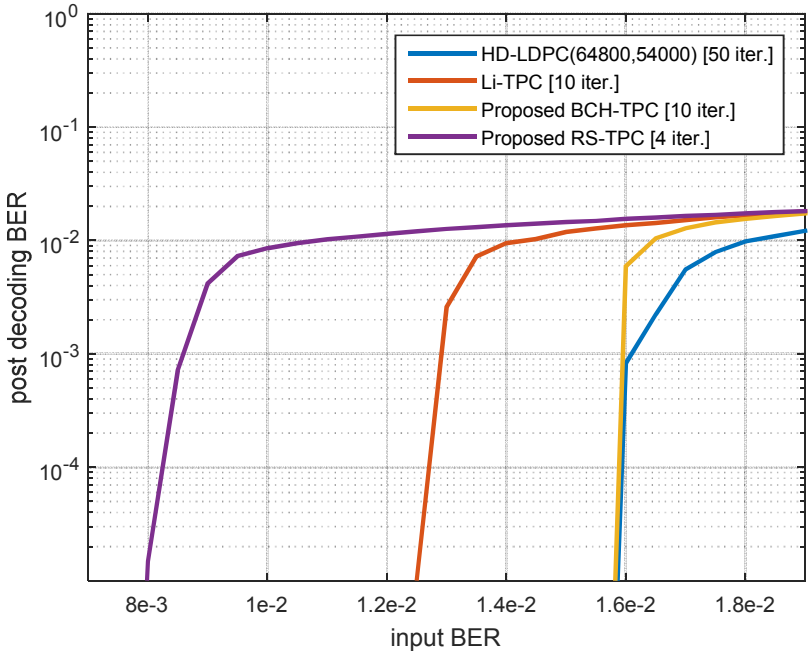


Figure 131: Comparison of error correction performance for HD-LDPC (blue), improved TPC (yellow), and the state of the art TPC (red).

The goal of the design of the proposed methods is not only good efficiency against single errors. In the targeted application, burst errors are more likely to occur than single errors[43]. Due to this reason, two more simulations against burst errors have been conducted. The simulations use error characteristics generated by the Markov chain, and error distributions of the characteristics are shown in Figure 132 (P[C] = 0.5, '*mixed errors*') and Figure 133 (P[C] = 0.9, '*burst errors*'). The characteristics are identical to the characteristics used in subsections 3.13.2 and 3.13.3, where RS, BCH, LDPC, and convolutional codes are compared. Figure 134 and Figure 135 depict the results of the simulations.

[43] See subsection 3.13.2.

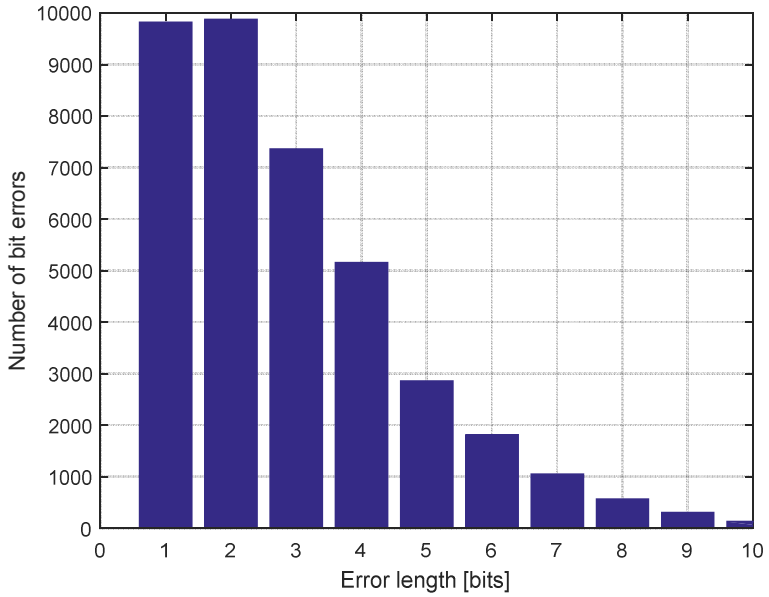


Figure 132: Error characteristic generated by the Markov chain with the C transition probability defined as P(C)=0.50 (mixed errors).

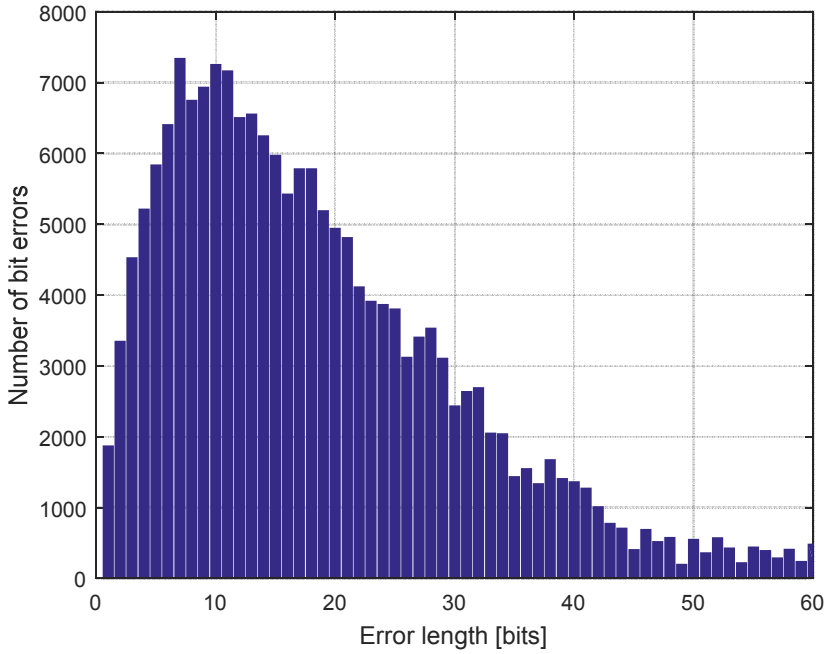


Figure 133: Error characteristic generated by a Markov chain with the C transition probability defined as P(C)=0.90 (burst errors).

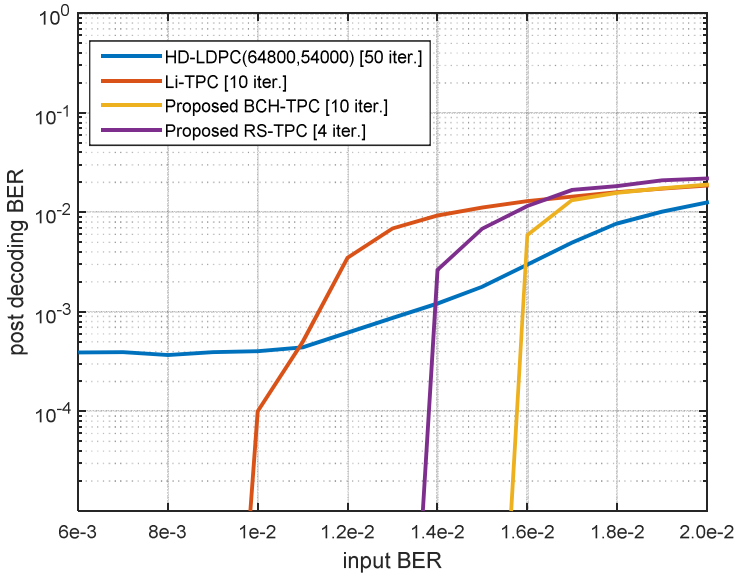


Figure 134: Comparison of error correction performance for HD-LDPC (blue), state of the art TPC (red), proposed BCH-TPC (yellow), and proposed RS-TPC (purple). Error characteristic generated by the Markov chain with P(C) = 0.5 (mixed errors) is considered.

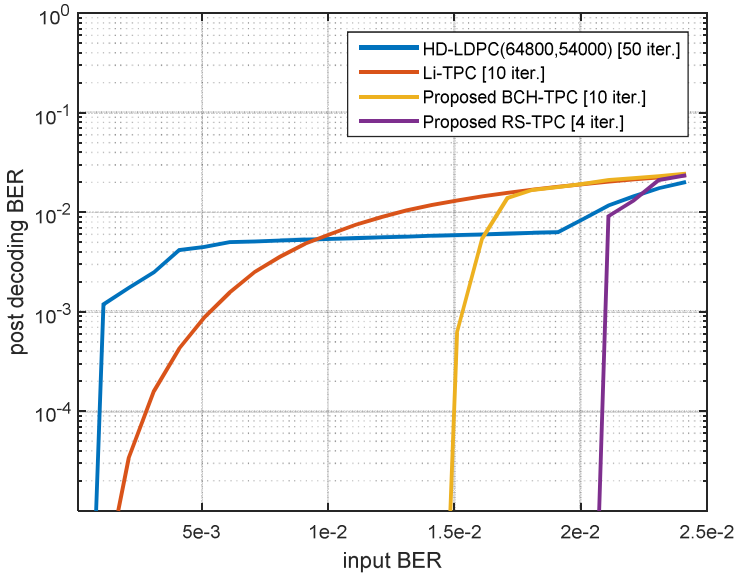


Figure 135: Comparison of error correction performance for HD-LDPC (blue), state of the art TPC (red), proposed BCH-TPC (yellow), and proposed RS-TPC (purple). Error characteristic generated by the Markov chain with P(C) = 0.9 (burst errors) is considered.

In both cases, the proposed methods are significantly better than the HD-LDPC and the Li-TPC methods. This is achieved due to the changed shape of the decoding code word. The diagonal part of the code word is not sensitive to burst errors and is resistant to stall-patterns[44] [91]. The other methods are blocked by burst errors and cannot operate without interleaving circuits in such conditions. The RS based decoder achieves the best results for burst errors, and if the number of iterations is increased, error correction performance is even more improved (Figure 136).

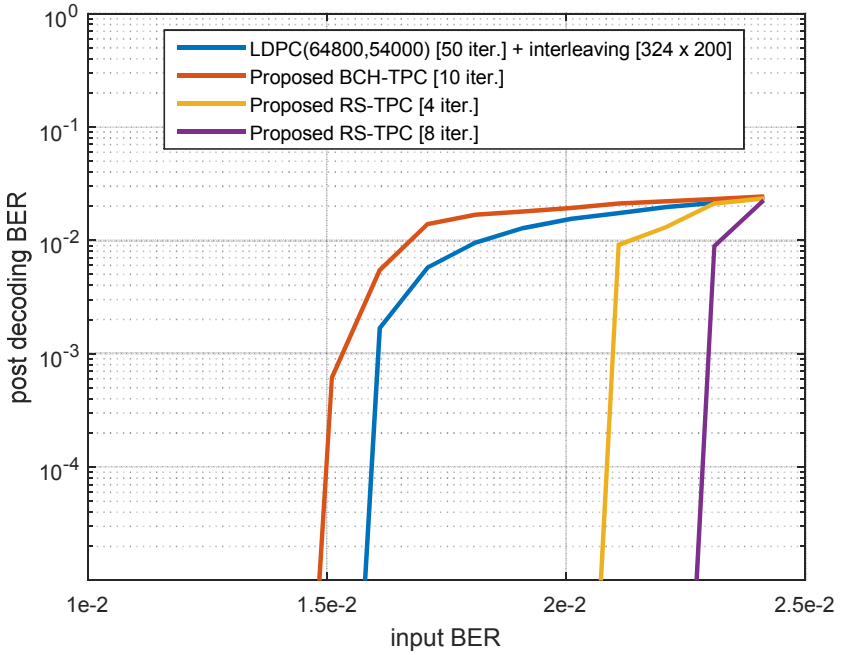


Figure 136: Comparison of error correction performance for HD-LDPC (blue), BCH-TPC (red), RS-TPC with four iterations (yellow), RS-TPC with eight iterations (purple). Error characteristic generated by the Markov chain with P(C) = 0.9 (burst errors) is considered. The HD-LDPC decoder is used with an external 324×200 matrix interleaver.

In such conditions, the HD-LDPC with interleaving performs significantly better than without the interleaver in the previous simulations (Figure 134 and Figure 135). All burst errors are converted to single errors before the decoding. The BCH-TPC cannot beat the HD-LDPC. Nevertheless, the proposed approach does not require any interleavers and provides almost the same performance for all conditions (single/burst errors). As expected, the RS-TPC outperforms all methods due to the burst-error correction performance of the RS component code. If the algorithm is run against long burst errors with eight iterations, the method corrects up to 50% more bit errors than the HD-LDPC. However, decoding latency of the RS-TPC method is

[44] See subsection 2.4.12.

very long comparing to LDPC decoders. In the considered implementation, four iterations of the RS-TPC require up to 800 clock cycles, while LDPC decoders presented in [75] require approx. 30-110 cycles. On the other hand, the HD-LDPC method uses a large matrix interleaver (320×200 memory elements). The interleaver requires extra decoding cycles and introduces additional decoding latency as well.

3.16.6 Required number of decoding iterations

In the previous section, error correction results of four different algorithms have been compared. All of them require an iterative decoding scheme. The optimal number of iterations is a tradeoff between consumed power, decoding latency, and error correction performance. In this subsection, the required number of decoding iterations to obtain errorless data at input BER = 1e-2 is investigated (all of the investigated algorithms can deliver error less data stream at the selected input BER value; the BER value corresponds to the sensitivity limit of common RF-transceivers). The BCH based methods and the HD-LDPC decoder are tested against single errors, and the RS-TPC against burst errors generated by the Markov chain with P(C) = 0.9. Therefore, all algorithms work in the targeted conditions. Figure 137 shows results achieved by the algorithms at input BER = 1e-2.

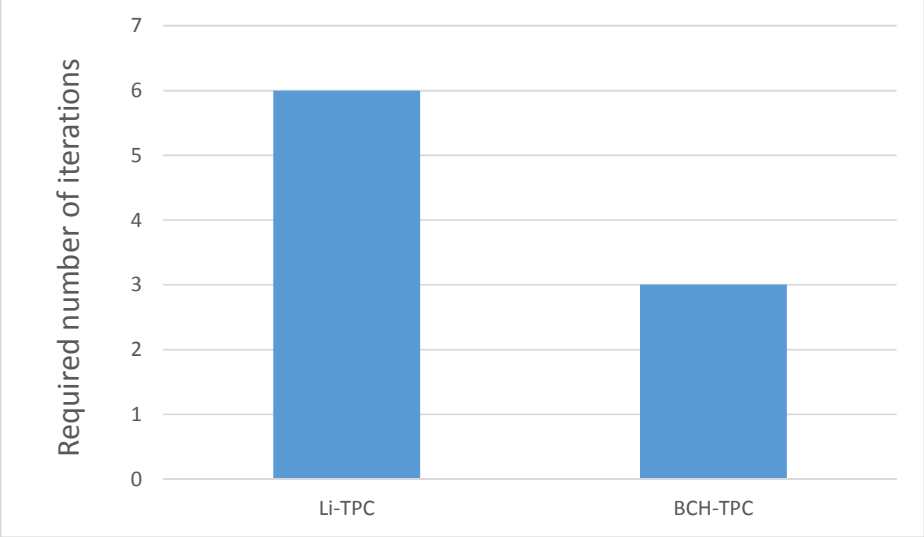


Figure 137: Required number of decoding iterations required by the tested algorithms to achieve an errorless code word at BER = 1e-2.

In this case, the Li-TPC decoder requires six iterations to correct all errors. The improved BCH-TPC solution achieves better decoding results and for the selected input BER = 1e-2 requires three iterations. In case of long burst errors, the RS-TPC can successfully decode the code word in two iterations only. The LDPC decoder requires 11 iterations to achieve an errorless data block. The HD-LDPC decoding

algorithm is based on belief propagation[45] and the decoding complexity cannot be directly compared to the TPC solutions (BCH- and RS-syndrome based decoding).

3.16.7 Estimation of decoding effort for BCH based TPC solutions

The BCH-TPC method uses BCH(511,466,t=5) component code, and the code obtains ~30% lower throughput than BCH(2047,2014,t=3) employed in the Li-TPC method. This suggests that the BCH-TPC requires ~30% more calculations and consumes ~30% more energy per bit. However, a detailed investigation of the methods shows that the newly proposed BCH-TPC method not only achieves higher error correction performance, but also requires less energy per bit. Peak energy consumption is higher, but average energy per bit is lower than for the Li-TPC method. This can be explained as follows: the Li-TPC method requires six iterations to achieve an errorless code word at input BER = 1e-2. In every iteration, 390 columns and 357 rows are decoded by BCH(2047,2014,t=3) component code decoder (127449 data bits). Thus, the decoder calculates 6×(390+357) = 4482 BCH code words per 127449 data bits.
In the newly proposed BCH-TPC method, the decoding matrix is smaller (232 × 232 = 53824 data bits), and a single code word covers a row and a column simultaneously. Thus, the decoder performs 232 BCH decoding operations per iteration. Moreover, the decoder requires three iterations instead of six, so 3 × 232 = 696 BCH decoding transactions are enough to decode a matrix with 53824 bits. Consequently, the newly proposed BCH-TPC method performs ~2.7 times less component code decoding operations per bit. Even if the method uses ~30% more complex component codes, the method performs ~270% less decoding attempts due to the increased error correction performance.

3.16.8 Hardware optimized BCH-TPC processing

The newly proposed TPC decoding method achieves higher coding gain and energy efficiency than the Li-TPC method proposed for 100 Gbps optical communication. However, implementation of the method for an ASIC device is relatively complicated. In the considered case, the method has to run at ≥100 Gbps. This is not achievable in the form of the algorithm presented in the previous sections due to decoding dependencies of the diagonal and horizontal parts of the code words. The BCH-TPC method can be calculated only in a sequential way by a single processing thread. Thus, there is no possibility to achieve high decoding throughput in ASIC/FPGA technology. In this subsection, modifications of the method reqiured for highly parallelized hardware implementation are explained. Technology aspects are considered according to Virtex7 technology, but all discussed technical issues apply for ASIC technologies as well.
Figure 138 shows a typical serial transceiver's processing path (GTH/GTX [98]). The GTH/GTX transceivers can be used to interface the FEC processor to the baseband and to the higher layers of the processing stack.

[45] The LDPC decoding based on belief propagation is explained in subsection 2.4.7.

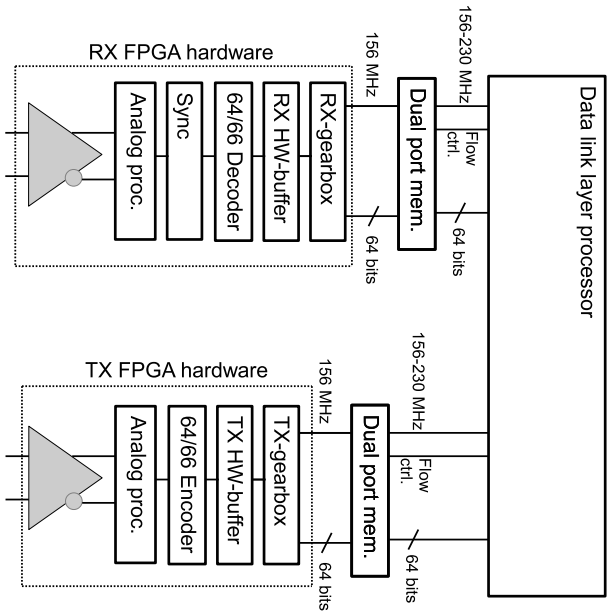


Figure 138: Hardware I/O architecture of the implemented data link layer processor.

The data refined by the serial transceiver is relayed to the FEC processor realized in FPGA or ASIC logic. The transceiver defines the interface for the processor implementation. From an efficiency point of view, the FEC processor interface has to be 100% compatible with the GTH/GTX interface according to the data bus size and clock frequency. If not, then data conversion methods have to be used. At the targeted data rate, processing time and consumed energy per bit is strictly limited. Thus, any data reformatting methods have to be avoided. It is necessary to find a solution, which allows to convert the size of the TPC processing matrix to common standards used in hardware (ASIC/FPGA). In this section, the TPC processing matrix is converted to a size that is compatible with 64-bit GTH/GTX serial transceivers. The same idea can be used for other bus sizes and technologies.
The native GTH/GTX interface provides 64 bits of data with every clock cycle at 156.25 MHz. The data processing clock can be adjusted in a range of 156.25-230.00 MHz using dual port memory buffers in FPGA/ASIC logic. In Figure 138, only a single RX and TX interface is shown (10 Gbps), and at least ten parallel instances are required to support 100 Gbps processing. Ten such structures deliver 640 data bits in each clock cycle to the FEC processor (at 156.25 MHz). The Li-TPC and BCH-TPC methods use matrix sizes of 390×390 and 232×277 bits respectively. The sizes are definitely not optimal for the hardware, and there is no possibility to interface it to the 64-bit GTH transceivers directly. Nevertheless, there is a possibility to change the matrix size and remove the decoding dependencies. In Figure 139, an optimized solution is presented. Instead of a single code word, two

component codes are used to remove decoding dependencies between the horizontal and diagonal parts of the BCH-TPC code words. Both codes are calculated over the horizontal-diagonal shapes, but the decoders are in counter-phases. For odd iterations, code *A* with redundancy data *A* is decoded. For even iterations, code *B* with redundancy bits *B* is calculated. In such case, the 64-bit data matrixes are iteratively decoded by the horizontal and diagonal BCH/RS code words. The decoding is similar to the proposed BCH-TPC procedure, and the 64×64 matrixes perfectly fit to the GTH/GTX interfaces. Additionally, there is no data dependency between the diagonal and horizontal code words like in the BCH-TPC. The effective code rate can be changed by modifying the horizontal length of the decoding matrix. The last advantage is the usage of the horizontal decoders with long pipelining (272 cycles in the presented case). Such architecture relaxes decoding timings, but error correction performance is insignificantly degraded (Figure 140).

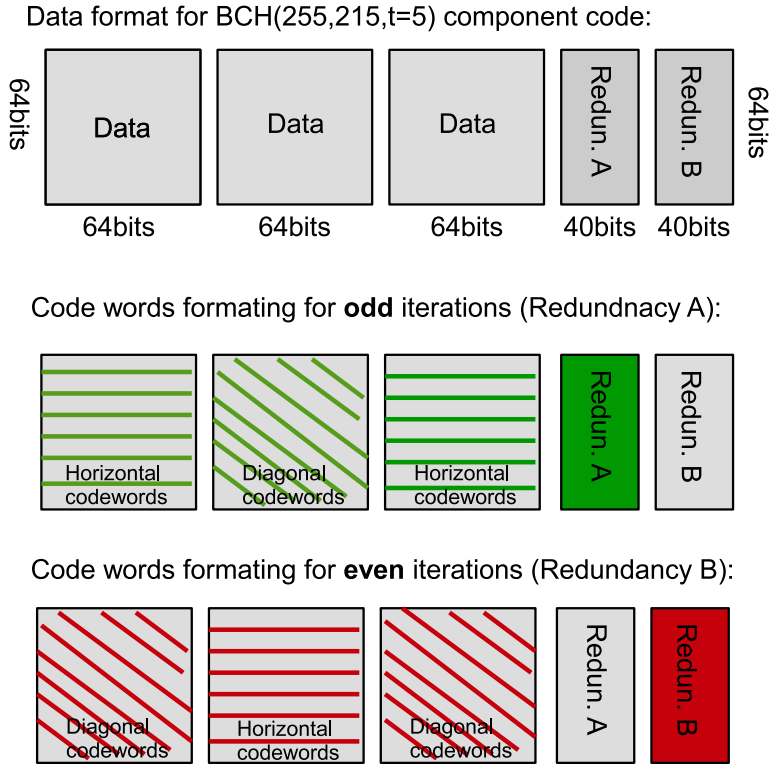


Figure 139: Modified version of the BCH/RS-TPC method for FPGA/ASIC implementations.

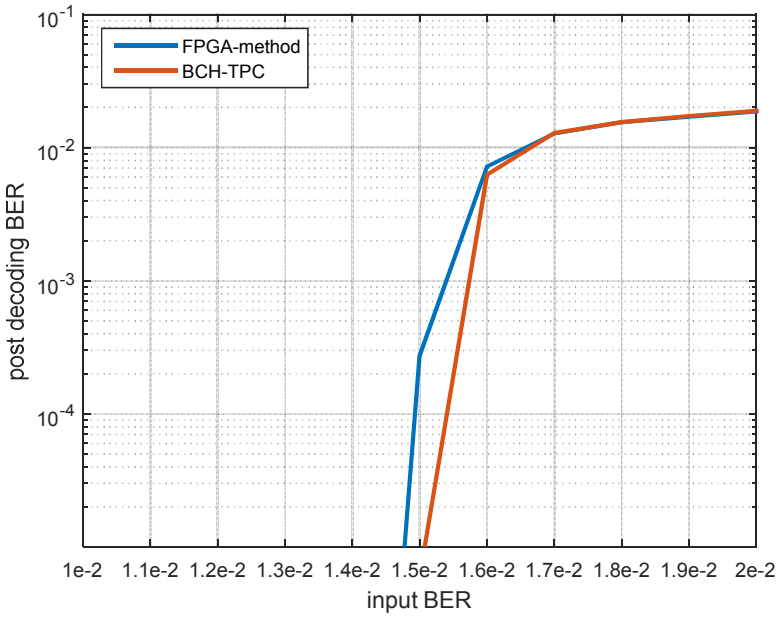


Figure 140: Comparison of error correction performance for the FPGA-modified version of the proposed TPC method (BCH 511,466,t=5).

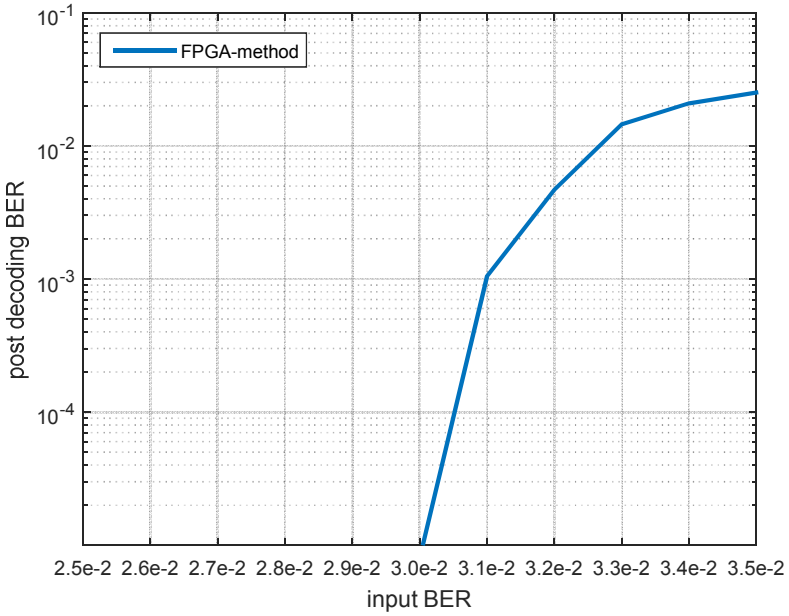


Figure 141: Error correction performance for the FPGA-modified version of the proposed TPC method (BCH 255,215,t=5).

The presented version of the algorithm shown in Figure 139 deals with BER ≈ 3e-2 at a code rate equal to R ≈ 0.8276 (Figure 141). It is more than required for the targeted 100 Gbps application (≤ 1e-2). Therefore, construction of TPC decoders based on BCH codes shorter than 511 bits with $t = 5$ is not considered in this work. Many different decoder architectures can be supported by changing the number of 64×64 data blocks and the base codes. The algorithm has all advantages of the BCH-TPC method. Decoding is usually finished in 2-3 iterations, is very effective against burst errors, and provides higher energy efficiency than the Li-TPC method. Moreover, the decoder does not require any external interleaving circuits, and can be based on RS and BCH component codes. The presented method allows to construct another than 64-bit interfaces as well.

3.17 Low latency FEC decoding for frame headers

If fragmentation and aggregation is in use, then the frame header is the most important part of the frame. Any single error that occurs in the fragmented payload reduces the goodput by 1/*n*, where the '*n*' is the number of data-fragments. The situation deteriorates greatly when a bit error occurs in the frame header. The header contains necessary information for frame decoding, e.g., the frame length and FEC encoding method. Without this information, the frame cannot be decoded and all '*n*' data-fragments are lost. Additionally, complexity of the frame-decoding pipeline is lower when the header is decoded immediately (without a delay). If information form the header is extracted immediately after receiving the header, the decoding pipeline can decode the frame-data without buffering. Thus, cache memory required for buffering the data can be avoided. For this purpose, an independent triple-modular redundancy decoder is proposed to improve the header decoding latency. A hardware implementation of the method decodes the header in time shorter than 5 ns. It means, information from the header can be used immediately and frame buffering is avoided. This is not possible when the header is encoded with RS or convolutional codes.

The triple-modular redundancy encoder sends the header three times, and the decoder checks the CRCs of the copies. If all copies have bit errors, then majority voting is performed on the copies, and the output of the voting is additionally checked. If at least one variant of four is correct, then the header is decoded successfully (Figure 142). The code rate of the presented method is equal to $R = 1/3$. In the considered case, the header is 20 bytes long and the encoded header uses 40 bytes of redundancy. The assumed frame length is not shorter than 64 kB, so the header overhead is not higher than 0.61 ‰ of the total frame length. Therefore, the overhead can be neglected.

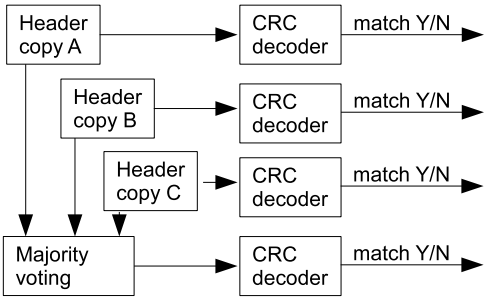


Figure 142: Triple-modular redundancy decoding circuit.

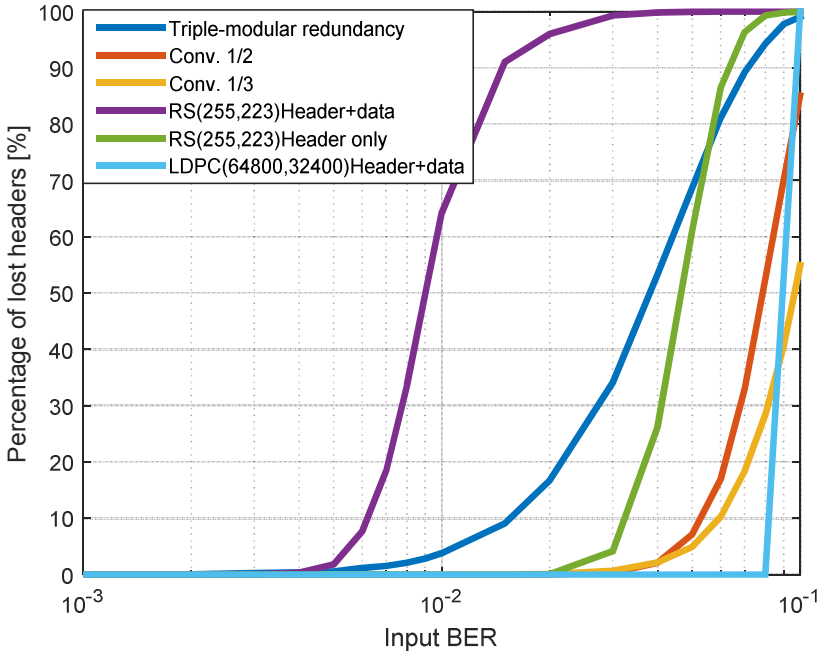


Figure 143: Comparison of error correction performance for different header coding schemes.

The introduced triple-modular routines are uncomplicated but the error correction performance is poor. In Figure 143, the algorithm is compared to the other previously mentioned FEC schemes (BCH, HD-LDPC, RS, HD-Convolutional). Convolutional coding with hard symbol representation corrects 2.5× higher input BER at the same code rate. If the gain is compared to RS(255,223), the results are more optimistic. If the header is excluded from the data block and encoded as a separate RS block, then the RS provides comparable correction performance (denoted as '*RS(255,223) Header only*' in Figure 143). It is difficult to estimate the difference and give an

accurate value in BER due to different slopes of the curves. However, when the header is encoded together with the frame data, then the triple redundancy corrects at least 4× higher input BER than the RS (denoted as '*RS(255,223) Header+data*'). Thus, the triple modular redundancy can be used as a substitute of the RS coding according to the correction performance. Moreover, the hardware implementation of the modular redundancy requires only 1 clock cycle (<< 5 ns) to provide decoding results. The RS needs ~450 clock cycles in the considered IHP implementation [107], and ~926 clock cycles in the Xilinx version of the IP-core [123]. Viterbi decoder provided by Xilinx requires at least 84 clock cycles to perform the same task [50].

3.18 Estimation of hardware resources required for 100 Gbps FEC decoder

Table 3 and Table 4 compare some common convolutional, LDPC, and RS coders implemented for FPGA and ASIC technologies. In the compared group of algorithms, RS consumes fewer hardware resources than the Viterbi and LDPC. It means, RS obtains higher goodput per a single logic cell (Table 4). This comparison shows the advantages of hard-decision RS decoders, but the analysis is not correct in this case. The algorithms are compared in typical configurations. For example, the code rate of the Viterbi decoder is defined as $R = 1/2$, and the code rate of the RS as $R = 239/255$. Thus, the algorithms have different error correction capabilities and such a study is not reliable. There are at least two possibilities to equate the algorithms' code rates. The Viterbi encoded stream can be punctured and the resulting code rate can be set closer to the RS decoder. However, most implementations of Viterbi decoders do not support such operating mode. The decoder usually accepts streams with a code rate in a range of 1/7 – 1/2 [50], [51], [55]. Alternatively, the RS code word can be shortened to achieve a code rate comparable to the Viterbi decoder. In this case, the required amount of additional resources required for the code shortening is insignificant in comparison to the total core size, but this approach also fails. The RS algorithm requires long decoding pipeline in most implementations. Shortening RS(255,239) and RS(255,223) codes to obtain $R = ½$ is usually not possible for common IP-cores [61], [71], [123], [124]. Additionally, changing the code rates of the codes may significantly influence the decoding efficiency, and such modified algorithms may be ineffective in terms of obtained coding gain and consumed hardware resources (this is investigated in section 3.15). Moreover, decoding performance depends on the error type on the decoder input, and both solutions prefer different error characteristics. Thus, comparison of the hardware resources is difficult and can lead to wrong conclusions. Therefore, in this section, a different benchmark is used. The goal is to estimate an average decoding goodput of typical decoders per 1 mm^2 of silicon, or per one FPGA look-up-table (LUT). This allows estimating how many FPGAs and how much silicon is required to implement 100 Gbps FEC engine. Due to this reason, Table 4 gives an approximation of average decoding goodput per single LUT of soft decision Viterbi and LDPC decoders in comparison to a hard decision RS(255,239) decoder.

The normalized goodput of the Xilinx-RS is 25 times higher than for the Xilinx-Viterbi, and approx. 700 higher than for the IHP-LDPC(1536,1152). If the Xilinx-Viterbi and IHP-LDPC solutions are scaled for a 100 Gbps system, the overhead is so high that it is not possible to fit the decoders into one of the high-end xc7vx690tffg1761-2 FPGAs. Moreover, the IHP-Viterbi decoder requires at least 23 FPGAs to support processing of 100 Gbps streams. If the LDPC coding is considered, the hardware overhead is even higher. Due to this reason, the Viterbi and LDPC decoders cannot be considered for FPGA implementation of the 100 Gbps FEC engine. Error correction performance of the IRS (interleaved Reed-Solomon) is limited but allows communicating over channels with BER ≈ 5e-3 for single errors, and BER ≈ 1e-2 for burst errors. This is almost enough to cover the entire operational range of a typical RF-transceiver. Sensitivity limit of wireless transceivers is usually defined as BER = 1e-2 [6], so the devices usually operate with lower BER. The IRS based FEC engine obtains moderate error correction performance for single errors, excellent performance for burst errors, fits into the Virtex7 device, and can be applied for ASICs with comparable performance to the Virtex7 FPGA (e.g., IHP 130 nm CMOS).

Table 5 compares LDPC and RS decoders [66], [107] synthesized into Infineon 40 nm CMOS technology. The IHP-RS(255,239) requires ~12 times less hardware resources than the soft decision IHP-LDPC(576,384) to achieve the same decoding goodput. Thus, IRS coding is one of assumed solutions to build a lightweight FEC engine for the targeted 100 Gbps application. Alternatively, modified BCH- and RS-based TPC solutions described in section 3.16 can be considered as possible improvements of the proposed IRS scheme.

If a high performance ASIC manufactured in ~28 nm technology is considered as the targeted technology, then soft decision LDPC decoders are recommended. Soft decision decoding improves error correction performance by ~2 dB comparing to the hard decision methods[46]. One of possible soft decision (SD) LDPC decoders constructed for the targeted data rate is shown in [75] and consumes 12 mm^2 in 65 nm SVT technology. Use of soft decision methods requires a fully integrated solution that integrates RF-frontend, baseband, ADC, and data link layer in a single mixed signals chip. Exchanging of the soft values requires at least 400 Gbps interconnections between the mentioned modules. Thus, all parts of the communication processor have to be integrated into a single package / single chip. Moreover, the ADC has to support multibit quantization (soft decision decoding). Currently, the targeted baseband does not have such possibility and supports hard decision FEC methods only. More details on soft decision FEC complexity can be found in section 6.1 (Appendix).

Convolutional codes require large interleavers for the considered PSSS-15 spreading and achieve poor error correction performance at the targeted code rate (~8/9). Thus, convolutional codes are not recommended for the DLL processor investigated in this work, neither for FPGA nor for ASIC implantations.

[46] According to Figure 42.

Implementation	Ref.	Max. Clk Freq. [MHz]	LUT area	FPGA/speed grade	Remarks
Xilinx Viterbi Decoder	[50]	286	2525	Virtex7/-1	3 bits soft coding; no interleaving; R=0.5; Constr. Len. 7; Traceback 96
Xilinx Viterbi Decoder	[50]	403	2525	Virtex7/-3	3 bits soft coding; no interleaving; R=0.5; Constr. Len. 7; Traceback 96
Creonic Viterbi Decoder	[55]	250	2984	Virtex6/-1	4 bits soft coding; no interleaving; R=0.5; Constr. Len. 7; Traceback 96
Creonic Viterbi Decoder	[55]	142	3054	Spartan6/-2	4 bits soft coding; no interleaving; R=0.5; Constr. Len. 7; Traceback 96
IHP Viterbi Decoder	[107]	170	12000	Virtex7/-2	5 bits soft coding; no interleaving; R=0.5; Constr. Len. 7; Traceback 96
Xilinx RS Decoder	[50]	294	765	Virtex7/-1	Hard coding; RS(255,239); 8 bits/clk; R=0.937
Xilinx RS Decoder	[50]	410	765	Virtex7/-3	Hard coding; RS(255,239); 8 bits/clk; R=0.937
Xilinx RS Encoder	[123]	388	260	Virtex7/-1	Hard coding; RS(255,239); 8 bits/clk; R=0.937
Xilinx RS Encoder	[123]	598	260	Virtex7/-3	Hard coding; RS(255,239); 8 bits/clk; R=0.937
IHP RS Decoder	[107]	285	2585	Virtex7/-2	Hard coding; RS(255,239); 8 bits/clk; R=0.937
IHP RS Encoder	[107]	457	200	Virtex7/-2	Hard coding; RS(255,239); 8 bits/clk; R=0.937
IHP RS Decoder	[107]	270	5554	Virtex7/-2	Hard coding; RS(255,223); 8 bits/clk; R=0.874

Table 3. Required hardware resources for selected forward error correction algorithms.

Implementation	Max. Clk Freq. [MHz]	LUT area	FPGA/ speed grade	Goodput [Mbps]	Goodput / 1 LUT [Mbps]
Xilinx Viterbi Decoder R=1/2 (3 bits quantization) [50]	403	2525	Virtex7/-3	403	0,16
Xilinx RS Decoder R=239/255≈0.87 (1 bits quantization) [64]	410	765	Virtex7/-3	3074	4,02
IHP LDPC Decoder R=3/4 (5 bits quantization) [66]	160	21682	Virtex7/-2	120	0,0053

Table 4: Forward error correction algorithms – achieved goodput in the Virtex7 FPGA.

Implementation	Ref.	Max. Clk Freq. [MHz]	Cell area [mm^2]	Technology	Goodput [Gbps]	Goodput / 1 mm^2 [Gbps/mm^2]
IHP LDPC(2304,1536) decoder	[66], [107]	445	0,79	Infineon 40 nm CMOS	1.774	2.2
IHP LDPC(576,384) decoder	[66], [107]	445	0,21	Infineon 40 nm CMOS	0.787	3.7
IHP RS(255,239) decoder	[66], [107]	540	0,088	Infineon 40 nm CMOS	4.050	46.0
IHP Viterbi decoder R=1/2	[66]	540	0,105	Infineon 40 nm CMOS	0.540	5.1

Table 5. Comparison of LDPC, Viterbi, and RS decoders (source: [66]*).*

3.19 Transmission statistics

The goodput is not only dependent on FEC and payload delivery-efficiency, but also on ACK-frames delivery-efficiency (ACK-length and error correction performance of the selected FEC code for the ACK-frames), number of PHY turnarounds, RF turnaround time, and timeouts. Other sources of the goodput degradation are bit errors that occur in frame headers. If the header contains errors, the whole frame (~64 kB) has to be discarded. In the header, the length of the frame, number of fragments, fragment size, and FEC method is defined. It is impossible to decode a frame without these details. Thus, the header has to be protected by a strong and low latency FEC as proposed in section 3.17.
Figure 144 and Figure 145 show transmission statistics of lost frame-fragments, frame-headers, and ACKs for the final data link layer protocol (the header is protected by a strong FEC code). The simulations include all approaches proposed in this work. In both cases, the goodput is limited by loss of data-fragments, not by loss of ACK-frames or frame headers. This confirms that the protocol operates correctly in the targeted input BER range [0; 1e-2].

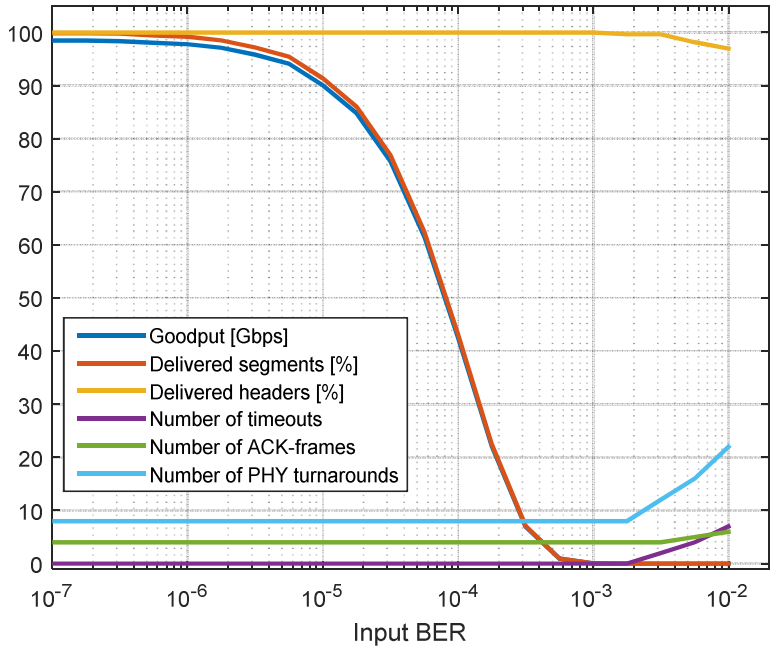


Figure 144: Communication statistics for transmission with deactivated FEC for data payload and activated triple-modular redundancy coding for frame headers.

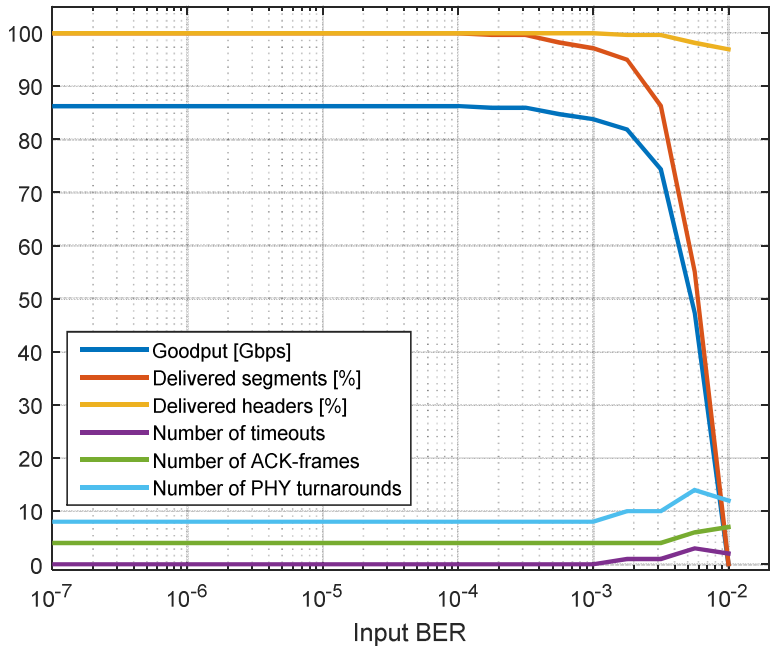


Figure 145: Transmission statistics for communication with activated RS(255,223) coding for data payload and triple-modular redundancy coding for frame headers.

4. Results

This chapter gives an overview of the proposed hardware architecture, and discusses energy efficiency of the implementation. All power and energy related aspects are considered in the context of IHP 130 nm and industrial 40 nm[47] CMOS technologies.

4.1 Data link layer accelerator hardware

Xilinx VC709 FPGA boards are used for the accelerator prototyping. Figure 146 shows the experimental setup based on Xilinx VC709 development kits [99].

Figure 146: DLL accelerator prototyping (Xilinx VC709). The presented FPGA set transfers data streams up to 120 Gbps (12 lanes[48]). SMA coaxial cables (80 Gbps; 8 lanes) and optical cables (40 Gbps; 4 lanes) with Markov chains are used as an emulated PHY medium.

The selected boards natively support four SFP+ cages suited for 10GBase-R Ethernet communication. The integrated FMC slot extends the boards by additional 32 SMA connectors. For this purpose, a third party FMC extension board is employed. All interfaces (SMA and SFP) use 10GBase-R Ethernet encapsulation.

[47] The details of the 40 nm technology cannot be published due to nondisclosure agreement (NDA) and license restrictions.
[48] The lane processing concept is explained in section 3.3.

4.2 Processing latency and goodput

The protocol implemented in the presented FPGA set (Figure 146) achieves goodput of ~97.5%. It means, the FPGA processes up to 116.4 Gbps of user data (goodput). Figure 147 shows the FPGA performance as a function of a channel BER. In the presented case, all available serial transceivers are used for data exchange. Therefore, the payload is generated by internal frame generators in the transmitter FPGA (TX-FPGA). Furthermore, the payload generators are removed from the design, and the Tilera is employed to supply the data to the FPGA. Therefore, the throughput per single FPGA board is reduced to 40 Gbps (4 lanes) due to the limited number of Ethernet interfaces.

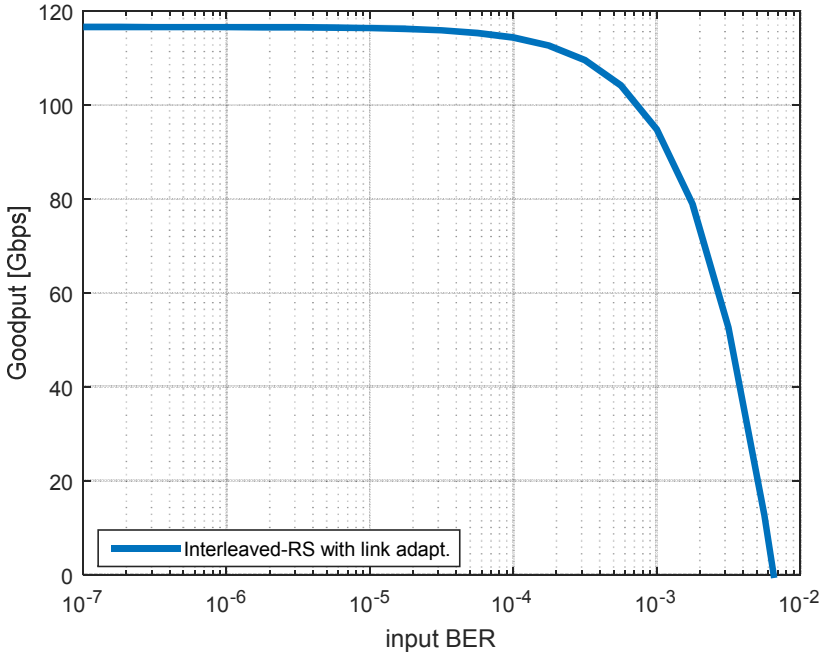


Figure 147: User data goodput as a function of the channel BER (12 processing lanes, HARQ-I, Interleaved Reed-Solomon coding with link adaptation, bit map ACK encoding, aggregation, and fragmentation).

The FPGA implementation introduces relatively long delay in the communication line. The receiver FPGA (RX-FPGA) requires 1219 clock cycles (~7.8 us) to forward data frames from the Tilera to the baseband. The transmitter works faster, and the data is delivered after 443 clock cycles. There are two sources of the delay in the RX-FPGA design. The employed IRS decoder requires up to 513 cycles to decode the first data block (255 cycles for block deserialization, up to 246 cycles for decoding, 12 cycles for additional pre- and post-processing).
The second issue is the Ethernet synchronization. The data scheduled for the transmission is initially stored in a FIFO, before Ethernet transmission is started.

This is required by aggregation and deaggregation modules of the Tilera frames. This leads to an additional delay of 384 cycles. To reduce the latency, two optimizations can be performed. Firstly, the IRS FEC engine can be replaced by a low latency FEC algorithm (e.g., by a fully unrolled LDPC decoder [75]). Secondly, Ethernet communication can be replaced by one of FPGA/ASIC dedicated solutions (e.g., Raw GTX/GTH or Aurora [98], [129]), or the whole demonstrator (Tilera functionality, DLL accelerator, baseband, RF-frontend) can be integrated in a single mixed-signal ASIC. Ethernet is a universal interface supported by many devices but adds significant latency to the communication. At a first glance, the overall FPGA latency of ~7.8 us is relatively long. However, when the value is compared to the Tilera processing time of ~2000 us [130], the FPGA delay is acceptable.

4.3 Implemented processor architecture

Only four 10GBASE-R Ethernet ports are available on the chosen VC709 development kit. Additionally, the kit has a possibility of using FMC-HPC [131] extension cards. On the FMC card, four additional Ethernets or eight GTH serial transceivers with 32 SMA connectors can be installed. In the final demonstrator, the embedded Ethernet ports are connected to the Tilera. The SMA ports, located on the FMC daughter card, will be used to connect to the baseband. This is explained in Figure 148.

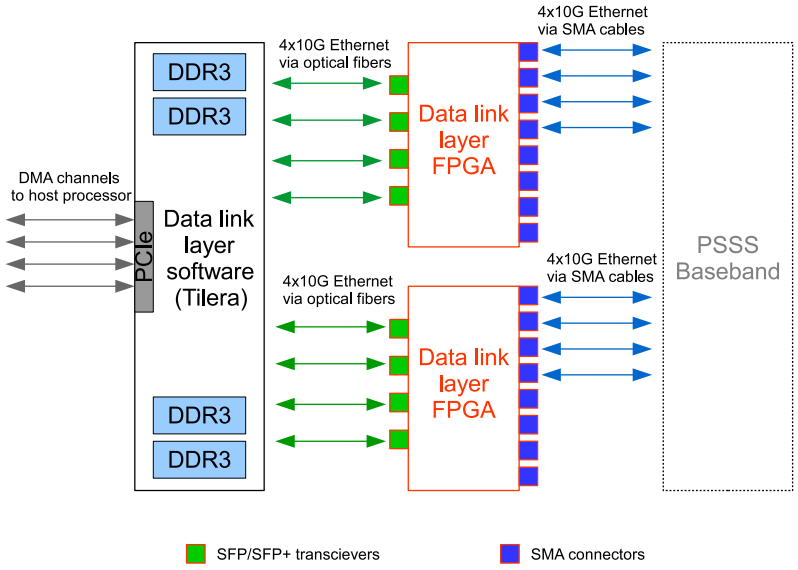


Figure 148: Architecture of the proposed and implemented 80 Gbps data link layer processor. The FPGA accelerator is marked in red. The Tilera card supports 4 DDR3 RAM memory modules, eight Ethernet interfaces (80 Gbps), and a single PCIe communication port.

It is possible to read and write up to 40 Gbps from/to the FPGA using the optical interfaces (10G Ethernet). This is the limitation of the VC709 board. To transfer 100 Gbps, at least three FPGA kits have to be used (up to four lanes can be processed by each VC709 FPGA). Generally, Virtex7 XC7VX690T FPGA chip supports up to 56 GTH channels (~560 Gbps) [104], but only few transceivers are provided on the VC709 PCB.

The Tilera TILEncore-Gx72 processor card (Figure 149), supports eight 10G Ethernet ports. Thus, the demonstrator's throughput is limited to 80 Gbps (according to Figure 148). The device includes 72 identical processor cores (tiles) interconnected by an on-chip network (Figure 150). Each tile consists of a 64-bit processor core, L1 cache, L2 cache, and a switch that connects the tiles to the network mesh. The device provides full-cache coherence among all cores. The embedded PCIe port supports data transfers up to 96 Gbps [105].

Figure 149: Tilera TILEncore-Gx72 processing card with eight 10G optical Ethernets and PCIe Tilera Gx-72 Processor. Figure retrieved from [105].

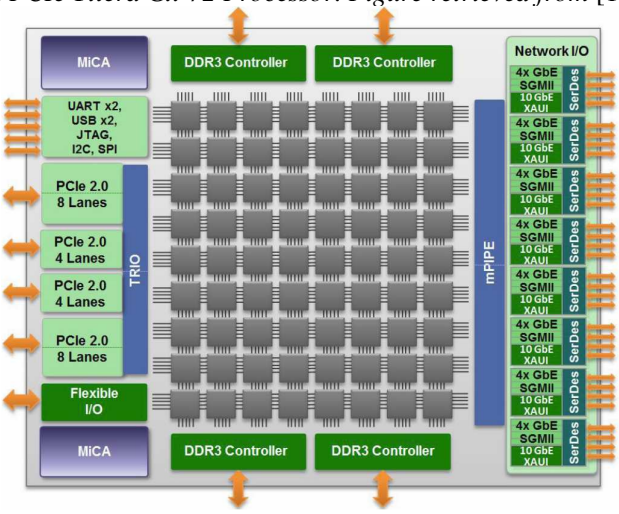


Figure 150: Block diagram of the Tilera Gx72 many-core processor. Figure retrieved from [105].

For more detail on implementation, see sections in appendix: 6.5 (Architecture of a single TX-lane), 6.6 (Architecture of a single RX-lane), and 6.7 (FPGA floorplan, resources, and clock domains). The main implementation issues are discussed in 6.1 (Soft and hard decision FEC processing), 6.2 (Comparison of high-speed serial protocols), 6.3 (Lanes deskewing), 6.4 (On chip flow controlling), and 6.8 (FPGA processing goodput).

4.4 130 nm and 40 nm CMOS technology results

This section introduces results of an ASIC synthesis. All FPGA technology depended components have to be removed from the design before the synthesis. It means, the accelerator has been examined without Ethernet cores. In such design, only one clock domain is necessary, and the clock frequency can be flexibly adjusted. The synthesis tool reports the maximal operational clock for a single lane up to 210 MHz and 900 MHz for IHP 130 nm and industrial 40 nm technologies respectively. Therefore, IHP technology requires eight lanes to support 100 Gbps processing (13.03 Gbps per lane). 40 nm technology enables faster processing (55.87 Gbps per lane) and two lane are enough to support 100 Gbps (Table 6).

Technology	Virtex7	130 nm IHP	40 nm
Coding	IRS(255,237)	IRS(255,223)	IRS(255,223)
Min. code rate	0.929	0.875	0.875
Max. goodput per lane	9.7 Gbps	13.03 Gbps	55.87 Gbps
Number of lanes	12	8	2
Max. goodput	116.4 Gbps	107.52 Gbps	111.74
Max. input BER (single errors)	~2e-3	~4e-3	~4e-3
Max. input BER (burst errors)	≥ 5e-3	≥ 1e-2	≥ 1e-2
Processing clock freq.	156.25 and 200 MHz (multiple clock domains)	210 MHz	900 MHz
ASIC area	71% of LUT resources available in Virtex7-690T	26.44 mm^2	0.81 mm^2

Table 6: Proposed lanes mapping for Virtex7, 130 nm IHP, and 40 nm technologies.

4.4.1 Synthesized chip area

Additional issue is synthesis of the RS entities. For the FPGA design, Xilinx optimized cores are in use, but for ASIC technologies a different solution has to be applied. In the IHP institute, two RS implementations are available [66], [121]. The first solution is a static coder that supports RS(255,239) coding. The second

implementation can additionally decode RS(255,223) and the coding can be switched on the fly with two redundancy steps: 16 or 32 symbols. It is not as flexible as the Xilinx solution, where the coding can be selected in 9 steps, but IHP implementation supports RS mode with 32 redundancy symbols per single RS block (RS 255,223). Thus, the IHP RS corrects almost double amount of errors compared to the Xilinx RS (Table 6). On the other hand, both IHP cores consume much more hardware resources than the dedicated Xilinx core. This is illustrated in Figure 151.

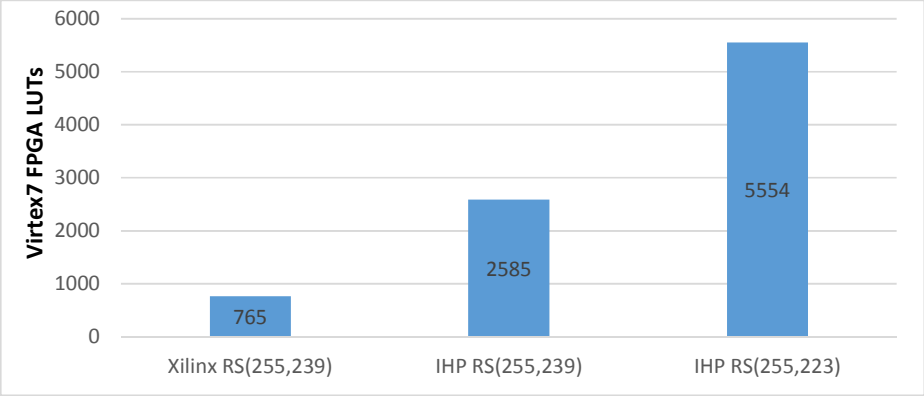


Figure 151: Comparison of FPGA resources required by Xilinx RS(255,239), IHP RS (255, 239), and IHP RS(255,223).

The comparison presented in Figure 151 suggests that the ASIC chip area occupied by the FEC engine will dominate. For the FPGA design, 66% of the resources are consumed by the FEC. For the ASIC, the value is even higher and is equal to ~92%. Figure 152 present the area of the synthesized ASICs with IHP-RS(255,223) cores.

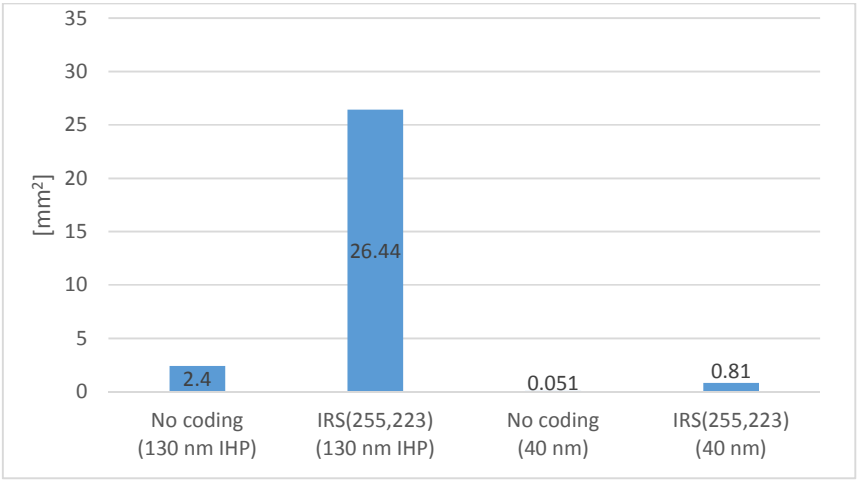


Figure 152: Estimated chip area occupied by the 100 Gbps accelerator synthesized into 130 and 40 nm technologies. Most the chip area is occupied by the FEC engine (~90%).

4.4.2 Power consumption

Consumed power and dissipated energy of the processor is a function of the channel BER. The RS computation effort dependents on the number of defected symbols in a block. Figure 153 shows consumed power as a function of the input BER for accelerators with IRS(255,239) and IRS(255,223) decoders.

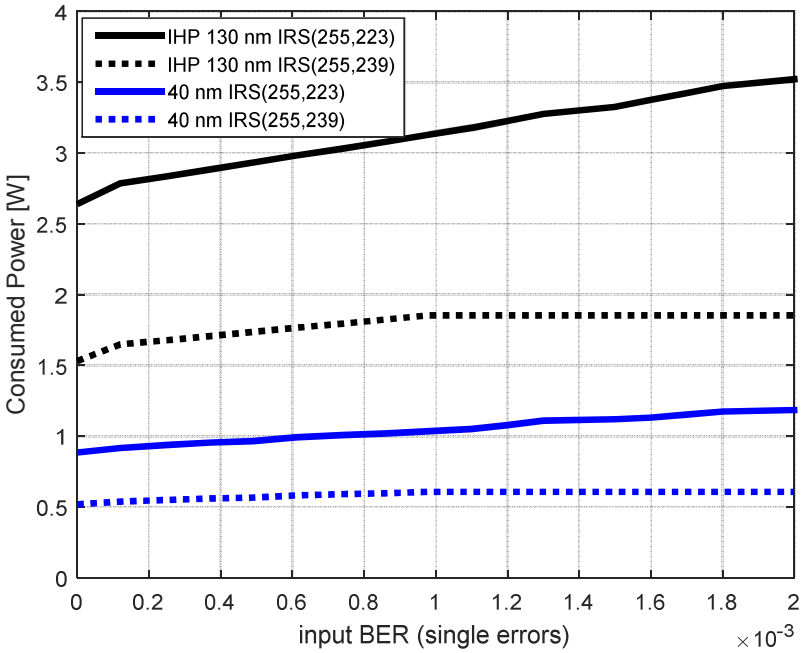


Figure 153: Power consumption as a function of the channel BER.

4.4.3 Consumed energy per data bit

Table 7 compares energy efficiency of the proposed IRS implementation to other published work. It is important to mention, that the proposed LDPC decoders operate at lower code rates. Thus, the LDPC codes can correct more bit errors (error correction performance is higher). Implementation of an IRS decoder with similar performance against single errors would consume more power than presented in the table. Additionally, both presented LDPC decoders support soft-decoded input data.

Algorithm	***LDPC***	***LDPC***	***RS***	***Interleaved RS***	***Interleaved RS***
Technology	65 nm GP	65 nm SVT	120 nm Philips	40 nm	40 nm
Code rate	0.5	0.8125 (13/16)	0.9372 (239/255)	0.9372 (239/255)	0.8745 (223/255)
Block size	672	672	2040	16320	16320
Quantization	5 bits	4 bits	1 bit	1 bit	1 bit
Energy Eff.	9 pJ/bit	37 pJ/ bit	6.5 pJ/ bit	6.08 pJ/ bit	12.9 pJ/ bit
Throughput	9 Gbps	161 Gbps	1.6 Gbps	119.6 Gbps	111.7 Gbps
Area	1.6 mm^2	12 mm^2	0.16 mm^2	0.55 mm^2	0.76 mm^2
Area Eff. [Gbit/s/mm^2]	5.63	13.4	10	217	147
Reference	[132]	[75]	[133]	This work	This work

Table 7: Comparison of selected LDPC, RS, and proposed IRS decoders.

4.5 Energy efficiency of link adaptation mechanisms

The FPGA Reed-Solomon cores cannot be synthesized into ASIC technologies, and detailed power profiling of the link adaptation mechanisms cannot be performed. Additionally, the RS FPGA netlist does not allow to power traces recording (license restrictions). Thus, only the ASIC-IRS cores can be considered for power estimation of the link adaptation mechanisms. However, the implementation supports only two IRS modes – IRS(255,239) and IRS(255,223). With two operating modes, the link adaptation cannot work efficiently. Nevertheless, there is a possibility to estimate consumed energy per bit for intermediate modes using the measured power values of the cores. The missing intermediate values can be evaluated using the number of executed operations for the IRS decoders. Table 8 and Figure 154 show a number of GF multiplications and additions required for syndrome based IRS decoding (worst case) as a function of the number of redundancy symbols [67], [134].

Module	**GF Multiplications**	**GF Additions**
Syndrome Computation	$2t(n-1)$	$2t(n-1)$
Key equation solver	$4t(2t+2)$	$2t(2t+1)$
Chien search	$n(t-1)$	nt
Forney's formula	$2t^2$	$t(2t-1)$
Total	$3nt+10t^2-n+6t$	$3nt+6t^2-t$

Table 8: Number of GF-multiplications and GF-additions required for syndrome-based IRS decoding (worst case) [67], [134].

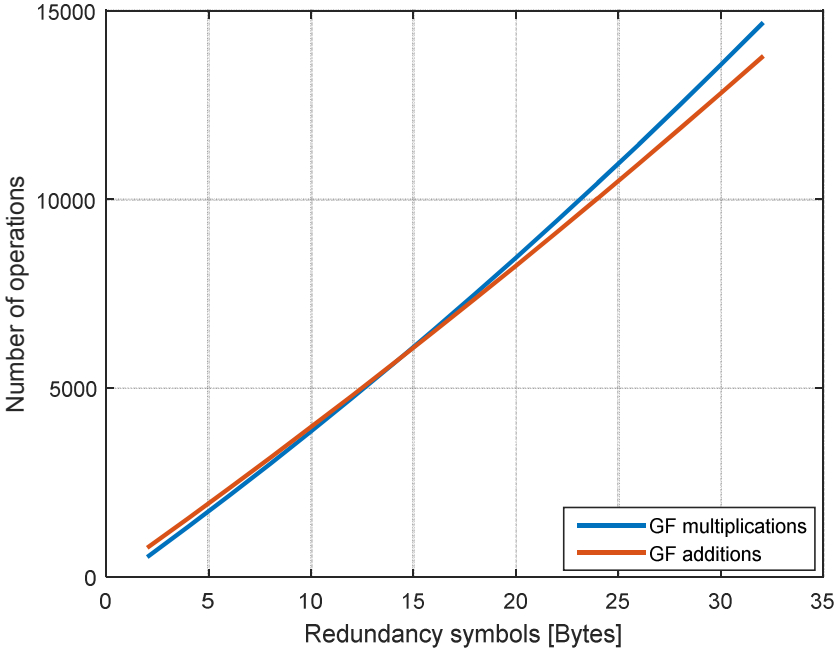


Figure 154: Number of GF-multiplications and -additions required for syndrome based RS decoder as a function of the number of redundancy symbols.

The energy efficiency estimation of the link adaptation has to include the worst-case decoding scenarios. The algorithm selects the lowest IRS mode, which can correct bit errors in the received stream. Briefly speaking, the mode with the lowest redundancy overhead is selected. Thus, the worst-case scenario is most likely to occur. Moreover, it is necessary to validate if the proposed methodology provides expected results for the measured values for RS(255,239) and RS(255,223). To check it, additional energy profiling is performed. RS(255,239) synthesized into the 40 nm technology requires 6.08 pJ of energy per data bit (pJ / bit), and according to Table 8 requires 6553 GF-multiplications and 6496 GF-additions. RS(255,239) synthesized into the same technology requires 12.9 pJ / bit, 14641 GF-multiplications, and 13760 GF-additions. It means, that RS(255,223) requires ×2.12 more energy per data bit, ×2.23 more GF-multiplications, and ×2.19 more additions. Table 9 summarizes the values:

	RS(255,239)	**RS(255,223)**	**RS(255,239) to RS(255,223) Ratio**
Multiplications	6553	14641	2.23
Additions	6496	13760	2.19
Energy per bit	6.08 pJ / bit	12.9 pJ / bit	2.12

Table 9: Comparison of consumed energy per bit and GF-operations for RS(255,239) and RS(255,223) IHP implementations (40 nm technology).

The measured energy ratio matches to the estimated complexity ratios of the decoders. Thus, this estimation has to be more or less correct and can be used for further investigations. In the presented case, the error of the estimation is lower than 6%, when the energy and multiplications ratios are compared. If additions are taken into the consideration, the error is even smaller (~3%). Nevertheless, from a complexity point of view, GF-multiplications are more complex than GF-additions. Thus, the number of multiplications has to be used with higher combining ratio than the number of additions. After that, energy per bit can be estimated for intermediate codes. This is summarized in Figure 155 and Table 10 (40 nm technology is considered[49]). Frame processing features (deaggregation, CRC, FSMs, etc.) consume relatively small amount of energy in comparison to the FEC decoding (frame processing: 0.656 pJ/bit; IRS decoding: up to 12.9 pJ/bit).

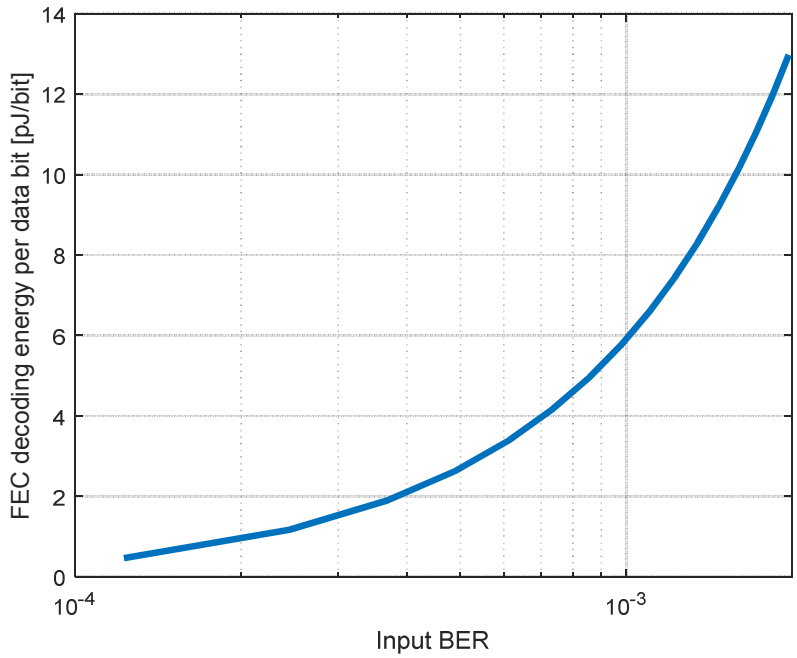


Figure 155: Required energy per processed data bit for IRS decoding with enabled link adaptation (40 nm technology is considered). Error characteristic with single errors is used to generate the graph.

[49] For more information about energy efficiency of the implementation synthesized into 130 nm IHP technology, see Appendix 6.9 - 6.12.

	RS(255,253)	RS(255,251)	RS(255,249)	RS(255,247)
Energy per bit	*0.46* pJ/bit	*1.17* pJ/bit	*1.89* pJ/bit	*2.63* pJ/bit
	RS(255,245)	**RS(255,243)**	**RS(255,241)**	**RS(255,239)**
Energy per bit	*3.39* pJ/bit	*4.18* pJ/bit	*4.96* pJ/bit	*5.77* pJ/bit
	RS(255,237)	**RS(255,235)**	**RS(255,233)**	**RS(255,231)**
Energy per bit	*6.60* pJ/bit	*7.45* pJ/bit	*8.31* pJ/bit	*9.19* pJ/bit
	RS(255,229)	**RS(255,227)**	**RS(255,225)**	**RS(255,223)**
Energy per bit	*10.09* pJ/bit	*11.01* pJ/bit	*11.95* pJ/bit	*12.9* pJ/bit
	Frame processing			
Energy per bit	*0.656* pJ/bit			

Table 10: Estimated energy per decoded data bit for intermediate codes (40 nm technology).

4.6 Eb/N0 and FEC energy

The designed data link layer presented in this work has to operate with other proposed DFG SPP1655 projects, especially with baseband and PHY-layer designed in the Real100G.COM and Real100G.RF projects. In the Real100G.COM scheme, the baseband uses parallel spread spectrum sequences (PSSS) with PAM-16 modulation (4 bits/Hz) and channel deconvolution. The PSSS and channel deconvolution provide some processing gain. This gain additionally improves transmission quality. Investigation of the PSSS and channel deconvolution is beyond the scope of this work. Thus, in this section a simplified BB and RF-frontend without the PSSS and channel deconvolution, but with PAM-16 modulation is considered. As a result, required Eb/N0 for the presented datalink layer processor is estimated.
Firstly, the required BER for the DLL implementation has to be defined. Due to the aggregation and fragmentation schemes, the DLL operates with deactivated FEC at BER ≈ 1e-6 with insignificant goodput degradation (goodput of 97.84% according to Figure 156). Improving the link quality above this value using FEC, or increasing transmission power does not improve the goodput significantly, but may lead to energy waste. Thus, from an energy efficiency point of view, the operational BER in range 1e-7 to 1e-6 is recommended to obtain the optimal goodput[50]. At BER ≈ 1e-5, the accelerator still operates and achieves goodput of 90.68%. However, BER values higher than ~1e-6 are not recommended, and in such case, the FEC effort has to be increased to reduce the number of retransmitted frames.
To estimate the required Eb/N0 it is necessary to investigate BER values for PAM-16 modulation with activated RS coding in the data link layer (Figure 157). The required Eb/N0 values can be read from the horizontal axis at BER = 1e-6 and BER = 1e-5. The values are presented in Figure 158 (40 nm technology is considered; results for 130 nm IHP technology are listed in Appendix 6.9 - 6.12).

[50] See section 4.7.

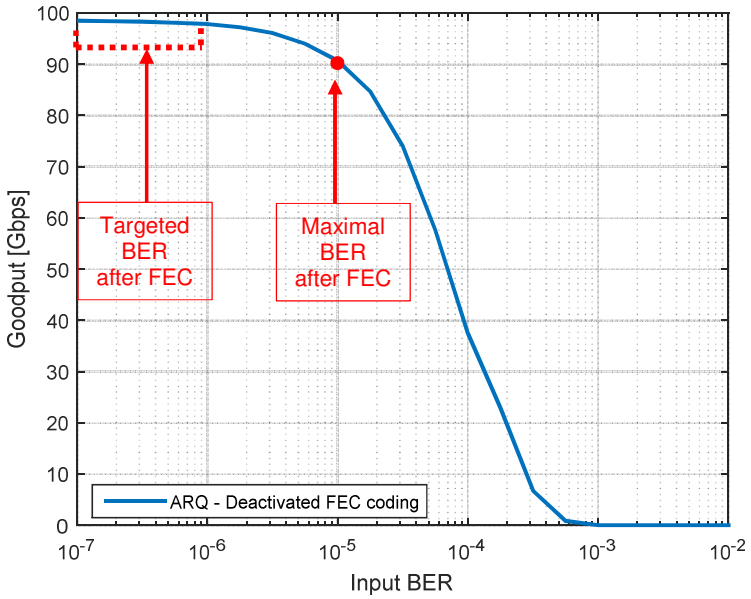


Figure 156: Goodput of the proposed DLL accelerator as a function of the channel BER (deactivated FEC).

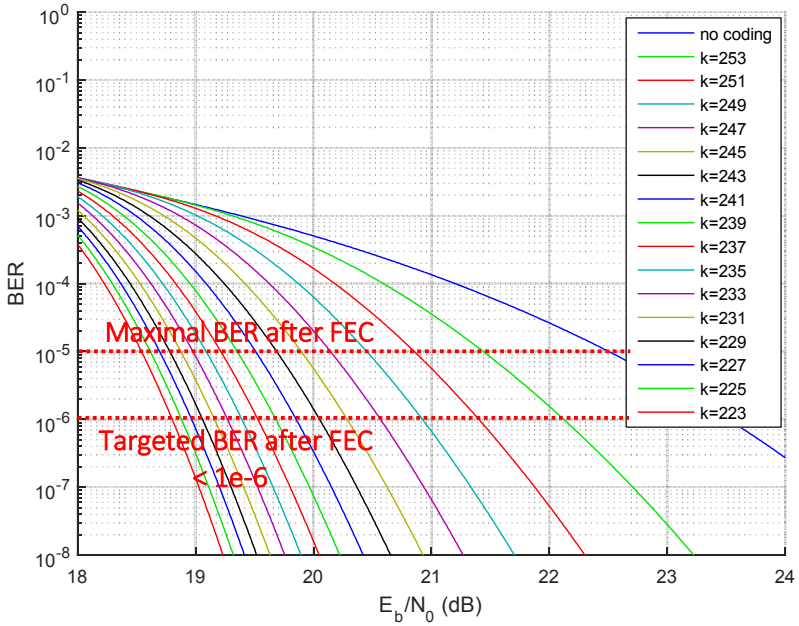


Figure 157: Reed-Solomon post decoding BER characteristics for PAM-16 modulation with RS(255, k) coding over an AWGN channel. The k parameter is in range 223 to 253 and this corresponds to RS(255,223) up to RS(255,253) coding.

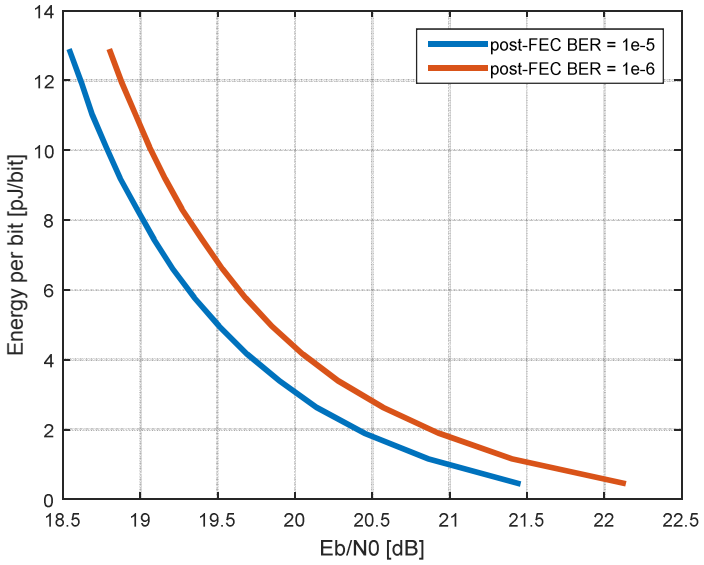


Figure 158: Energy per data bit required by the FEC processor to support the required post-FEC BER: 1e-5 denoted in blue; 1e-6 denoted in red (40 nm technology).

The measurements in Figure 158 correspond to the following goodput characteristics of the DLL processor (Figure 159):

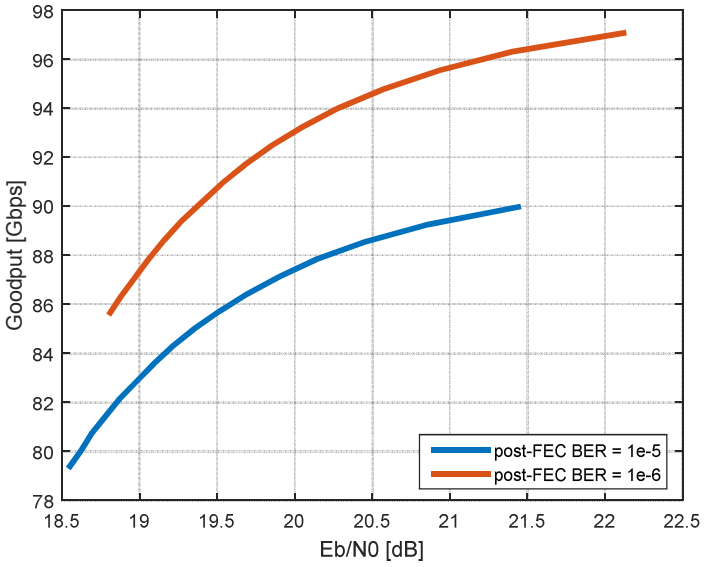


Figure 159: Estimated goodput for the selected post-FEC BER values: 1e-5 denoted in blue; 1e-6 denoted in red.

The goodput (Figure 159) decreases with the decrease of the EB/N0, even if the post-FEC BER values are constant (1e-6 and 1e-5). This is caused by the link adaptation approach but not by retransmissions (ARQ). The processor increases the number of redundancy bits with the decrease of the EB/N0. Thus, the average amount of user data in a frame decreases, so the goodput decreases as well.

4.7 ARQ and FEC tradeoff

One of the most interesting aspects considered in this work is comparison of ARQ and FEC in context of consumed energy per bit. Section 3.12 proposes an algorithm to find a tradeoff between FEC and ARQ to maximize average goodput (link adaptation). In this section, a similar study is described, but instead of optimization of goodput, consumed energy per bit is considered. The goal is to find a tradeoff between ARQ and FEC, so the average energy per bit is minimized. Moreover, the estimation includes not only DLL processing but also energy consumed by the baseband and RF-frontend. Firstly, a simulation model required for the estimation is introduced.

4.7.1 Simulation model

The estimation of energy required to transmit a single bit has to include complete transceiver processing (that is data link layer, baseband, and RF-transmission). Otherwise, the estimation can be unreliable. Each retransmission of a lost frame requires not only doubled processing energy at the data link layer, but also doubled energy for frame transmission and baseband processing. Thus, the data link layer has to include the lower layers into the account (on RX and TX sides – Figure 160). According to these requirements, and to the BER of the channel, the optimal FEC code has to be selected.

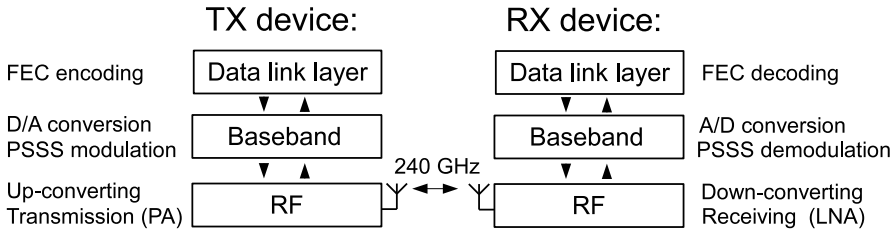


Figure 160: Overview of a communication system with the main sources of power dissipation.

The TX-device model has to include energy required for FEC encoding, fragmentation, aggregation, PSSS modulation, digital to analog conversion (D/A), RF up-converting, and amplification of the signal at the carrier frequency (PA).

Additionally, the RX-device dissipates power used for amplification of the received RF signal (LNA), down-converting, PSSS demodulation, analog to digital conversion (A/D), deaggregation, and FEC decoding. When a frame is lost during transmission, then each bit of the lost frame uses doubled amount of energy for the mentioned processes. The consumed energy in the system can be summarized by the following equation (5.1):

$$E \approx \left(DLL_{TX} + \frac{BB_{TX} + RF_{TX} + RF_{RX} + BB_{RX}}{FE \times CR} + DLL_{RX}\right) \times \frac{1}{1 - SER} \quad (5.1),$$

where:

E - estimated energy,
DLL_{TX} - energy required by the data link layer of the transmitter,
BB_{TX} - energy required by the baseband of the transmitter,
RF_{TX} - energy required by the RF-frontend of the transmitter,
DLL_{RX} - energy required by the data link layer of the receiver,
BB_{RX} - energy required by the baseband of the receiver,
RF_{RX} - energy required by the RF-frontend of the receiver,
FE - frame efficiency defined as: *'payload / (payload + overhead)',*
CR - code rate of the selected FEC code,
SER - fragment ('segment') error rate.

4.7.2 Energy efficiency of the IRS encoder

Up to this time, the energy consumed by IRS encoder and DLL transmitter was not considered in this work. The FEC encoder usually is less complex than the decoder. Thus, it was not critical to estimate the computation complexity for the encoders. However, to estimate the system's energy efficiency it is necessary to perform power profiling for the complete transmitter and receiver. The IRS encoder is controlled by link adaptation algorithm, thus the consumed power depends on BER (Figure 161, 40 nm technology is considered[51]). Table 11 gives accurate values for intermediate codes used by the link adaptation. Furthermore, Figure 162 shows decoder to encoder energy per bit ratio as a function of symbol error correction capability. The decoder requires approx. 2 to 4 times more energy than the IRS encoder.

[51] For more information about energy efficiency of the implementation synthesized into 130 nm IHP technology, see Appendix 6.9 - 6.12.

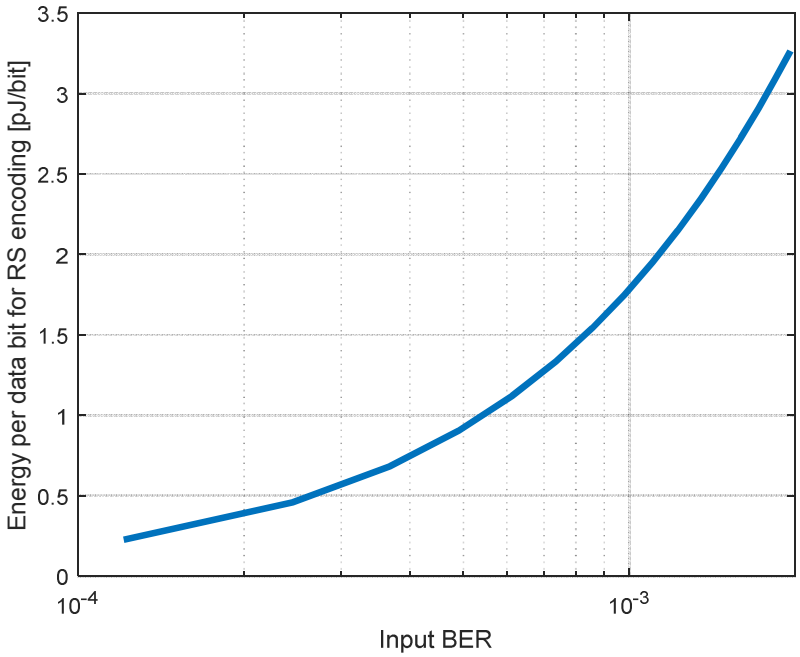


Figure 161: Energy per bit required by the implemented RS encoder as a function of BER (link adaptation, 40 nm technology).

	RS(255,253)	**RS(255,251)**	**RS(255,249)**	**RS(255,247)**
Energy per bit	*0.23 pJ/bit*	*0.45 pJ/bit*	*0.68 pJ/bit*	*0.9 pJ/bit*
	RS(255,245)	**RS(255,243)**	**RS(255,241)**	**RS(255,239)**
Energy per bit	*1.12 pJ/bit*	*1.33 pJ/bit*	*1.54 pJ/bit*	*1.75 pJ/bit*
	RS(255,237)	**RS(255,235)**	**RS(255,233)**	**RS(255,231)**
Energy per bit	*1.95 pJ/bit*	*2.15 pJ/bit*	*2.34 pJ/bit*	*2.53 pJ/bit*
	RS(255,229)	**RS(255,227)**	**RS(255,225)**	**RS(255,223)**
Energy per bit	*2.72 pJ/bit*	*2.9 pJ/bit*	*3.08 pJ/bit*	*3.26 pJ/bit*
	Frame processing			
Energy per bit	*0.55 pJ/bit*			

Table 11: Estimated energy per processed data bit for intermediate codes (RS encoder and DLL on the transmitter side).

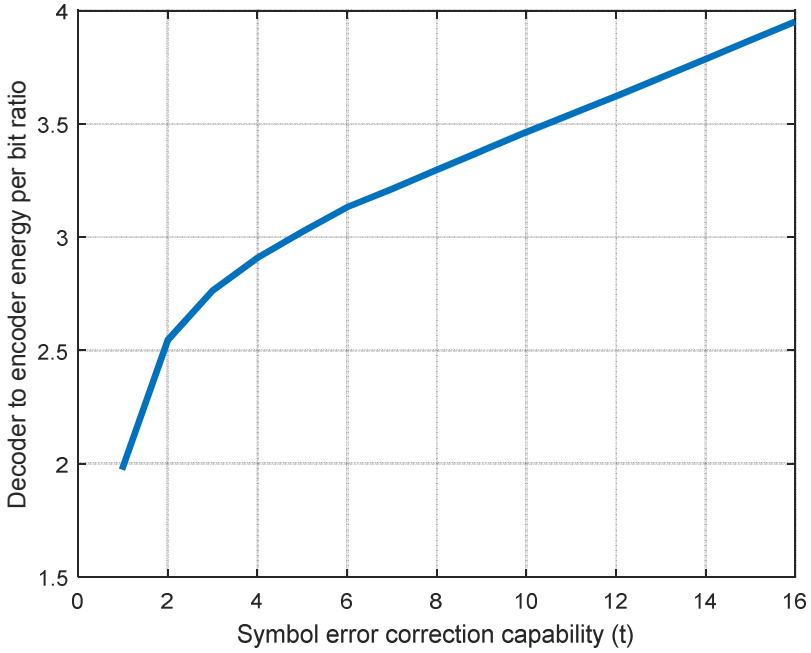


Figure 162: IHP-RS decoder to encoder energy per bit ratio as a function of symbol error correction capability.

4.7.3 Methodology

The baseband and the RF-frontend ASICs are not released yet, and energy consumed by the modules cannot be estimated. Currently, only the DLL-accelerator is examined according to the consumed power. Moreover, the energy dissipated by the Tilera card is unknown. Thus, it is not possible to give an accurate value of the consumed energy per bit for the whole system. Nevertheless, it is possible to define some energy strategies and energy thresholds when the corresponding strategies are applied. The DLL processor can influence the coding according to the expected QoS (quality of service). Therefore, the processor has to decide if coding has to be increased or decreased. The amount of energy that can be saved applying less complex FEC cannot be wasted by recurring retransmission. Thus, the main aspect is the balance between retransmissions (ARQ) and FEC. Aggressive FEC coding requires more energy per bit for computations and reduces the effective code rate of the transmitted stream due to redundancy bits. This is expressed by the '*CR*' value in formula 5.1 ($CR \leq 1$). Bit retransmission requires double energy for encoding and decoding, as well double energy for receiving and transmitting. This means, retransmissions are relatively expensive from the energy point of view. Moreover, Table 12 compares four radios according to consumed energy. For the considered radios, transmission and receiving of a single bit costs approx. 692 pJ/bit to 316800

pJ/bit. Energy required for IRS processing varies between ~0.69 pJ/bit to ~16.16 pJ/bit (encoding + decoding). Thus, retransmission of a bit is more expensive than FEC coding. In the future, new technologies and higher data rate will allow to reduce the consumed energy per transmitted bit, so the values given in Table 12 cannot be considered for the targeted 100 Gbps transceivers. On the other hand, also the FEC methods used in the DLL processor can be significantly improved. For example, using methods presented in [75], [132], [133] and employing a newer technology than 40 nm CMOS.

To find the best energy efficiency per bit, the model has to select a FEC mode with the highest goodput, and compare the mode to another mode with reduced FEC. Such estimation is performed for arbitrary selected BER values (4e-3, 3e-3, 2e-3, 1e-3, 1e-4, 1e-5, 1e-6, and 1e-7). This operation can by expressed by the following formulas:

$$x = BB_{TX} + RF_{TX} + RF_{RX} + BB_{RX} \quad (5.2),$$

$$E_1 \approx \left(DLL_{TX1} + \frac{x}{FE \times CR1} + DLL_{RX1}\right) \times \frac{1}{1-SER1} \quad (5.3),$$

$$E_2 \approx \left(DLL_{TX2} + \frac{x}{FE \times CR2} + DLL_{RX2}\right) \times \frac{1}{1-SER2} \quad (5.4),$$

The model has to find x_1 value defined by (5.5):

$$\bigvee_{x_1} \bigwedge_{0<x<x_1} E_1 > E_2 \quad (5.5),$$

where:

- x – energy per bit consumed by the baseband and RF fronted on the TX and RX sides,
- x_1 – energy boundary for the 'x', so the condition '$E_1 > E_2$' is satisfied
- E_1 – estimated energy for the mode with the maximal goodput at the targeted BER,
- E_2 – estimated energy for the alternative mode with reduced FEC at the targeted BER.

Other variables have the same meaning like in formula (5.1). The model searches for x_1 value that defines the energy per bit threshold for the 'x' value (energy per bit consumed by the baseband and RF-fronted on the TX and RX sides – equation 5.2). Briefly speaking, the model estimates when decreasing the FEC and increasing ARQ makes sense, and defines energy boundaries ('x_1') for the baseband and RF-frontend when reduction of the FEC is advantageous.

Chip	Functionality	TX current	RX current	Data rate	RF power	Supply Voltage	Energy per TX bit	Energy per RX bit	TX efficiency (transmitted power to consumed power)	Reference
CC1101	Low-cost, Low-power Sub 1 GHz Baseband and RF-frontend	34 mA	14 mA	0.5 Mbps	12 dBm	3.3 V	224400 pJ / bit	92400 pJ / bit	~14.12 %	[6]
WL1837	802.11n Baseband and RF-frontend	210 mA	85 mA	130 Mbps	12 dBm	3.3 V	5300 pJ / bit	2200 pJ / bit	~2.28 %	[135]
SKY85803	802.11ac RF-frontend	332 mA	12 mA	780 Mbps	13 dBm	3.3 V	1400 pJ / bit	50 pJ / bit	~1.82 %	[136]
Real100.RF TU Wuppertal	RF-frontend (240 GHz)	---	---	2730 Mbps	-4.4 dBm	---	375 pJ/bit	317 pJ/bit	~0.11 %	[120]

Table 12: Comparison of selected state of the art RF-transceivers.

4.7.4 Results

Results presented in this subsection (Tables 15–22) should be interpreted in the following way (40 nm technology is considered[52]). Column denoted as "*x* value [pJ/bit]" indicates one of the codes as the "*optimal mode*" (the mode with the highest goodput). The values with '<' mark indicate the required energy per bit for the baseband and RF frontend, so that the corresponding mode will be more energy efficient than the "optimal mode" ('x_1' value according to formula 5.5). A code denoted as "$x_1 \notin R+$" means that the corresponding mode cannot be more efficient than the "*optimal mode*" under all '*x*' conditions. In the considered cases, this is caused by too high FEC overhead for the assumed input BER.
The most interesting situation occurs for input BER = 3e-3 (Table 14). RS(255,225) is the mode that achieves the highest goodput at the considered BER (denoted as the "*optimal mode*"). If the baseband and RF-processing is reduced below 555 pJ/bit, the RS(255,227) will obtain higher energy efficiency than RS(255,225). A value below 555 pJ/bit cannot be achieved by transceivers shown in Table 12. Currently, 240 GHz RF-frontend at 2.73 Gbps and -4.4 dBm presented in [120] consumes approx. 375 pJ/bit in TX mode and 317 pJ/bit in RX mode (692 pJ/bit without baseband). The maximal power of the power amplifier (PA) is very low, so the PA consumes relatively small part of the energy consumed by the transmitter. If the power amplifier's output power will be increased, then also the consumed energy per bit will increase.
For the targeted 40 nm technology and proposed IRS codes, the energy required for FEC computation is lower than energy required for data transmission. This means, the processor has to optimize the goodput by adopting the code rate of the produced stream. Therefore, the proposed link adaptation algorithm (section 3.12) optimizes not only goodput but also energy efficiency per bit.

BER = 4e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,223)	83.678	Optimal mode	1.0451
RS(255,225)	79.84	< 3	1.1052
RS(255,227)	72.467	< 1	1.2284

Table 13: Results of the mathematical model (5.5) for BER = 4e-3.

BER = 3e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1 / (1 - *SER*) (bit retransmission ratio)**
RS(255,223)	87.401	$x_1 \notin R+$	1.0076

[52] For more information about energy efficiency of the implementation synthesized into 130 nm IHP technology, see Appendix 6.9 - 6.12.

RS(255,225)	87.42	Optimal mode	1.0018
RS(255,227)	87.286	< 555	1.0199
RS(255,229)	86.023	< 94	1.044
RS(255,231)	81.471	< 24	1.1119

Table 14: Results of the mathematical model (5.5) for BER = 3e-3.

BER = 2e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,227)	88.986	$x_1 \notin R+$	1.0004
RS(255,229)	89.634	$x_1 \notin R+$	1.0019
RS(255,231)	90.078	$x_1 \notin R+$	1.0057
RS(255,233)	90.123	Optimal mode	1.0139
RS(255,235)	88.903	< 53	1.0366
RS(255,237)	83.696	< 13	1.1105

Table 15: Results of the mathematical model (5.5) for BER = 2e-3.

BER = 1e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,237)	92.916	$x_1 \notin R+$	1.0003
RS(255,239)	93.394	Optimal mode	1.0035
RS(255,241)	92.912	< 165	1.0172
RS(255,243)	90.051	< 41	1.0582
RS(255,245)	78.285	< 8	1.2273

Table 16: Results of the mathematical model (5.5) for BER = 1e-3.

BER = 1e-4			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,245)	96.078	$x_1 \notin R+$	1
RS(255,247)	96.863	$x_1 \notin R+$	1
RS(255,249)	97.494	Optimal mode	1.0016
RS(255,251)	95.83	< 141	1.0271
RS(255,253)	80.555	< 23	1.2316

Table 17: Results of the mathematical model (5.5) for BER = 1e-4.

BER = 1e-5			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,251)	98.431	$x_1 \notin R+$	1
RS(255,253)	98.604	Optimal mode	1.0062
no coding	79.806	< 5	1.253

Table 18: Results of the mathematical model (5.5) for BER = 1e-5.

BER = 1e-6			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,253)	99.216	Optimal mode	1
no coding	96.94	< 33	1.0316

Table 19: Results of the mathematical model (5.5) for BER = 1e-6.

BER = 1e-7			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,253)	99.216	$x_1 \notin R+$	1
no coding	99.388	Optimal mode	1.0062

Table 20: Results of the mathematical model (5.5) for BER = 1e-7.

4.8 FEC and output power tradeoff

The proposed IRS coding achieves up to 4.75 dB coding gain (Figure 163, in case of an AWGN channel the gain achieved by the IRS can be approximated by the gain obtained by plain RS codes). The gain costs up to ~16.2 pJ/bit of energy. This section compares energy consumed by power amplifiers (PAs) from four solutions working in the terahertz band (Table 21). The main research idea is to check, whether coding gain obtained by IRS codes costs less than increasing the output power of the considered PAs according to the consumed energy per bit.

	Frequency	**P_{out_max}**	**PAE**	**Ref.**	**Institution**
PA-1	250 GHz	+10 dBm	1% – 3%	[137]	KIT Karlsruhe
PA-2	240 GHz	-4.4 dBm	~0.11%	[120]	TU Wuppertal
PA-3	200 GHz	+7.4 dBm	0.5% – 2.5%	[138]	IAF Freiburg
PA-4	94 GHz	+20 dBm	2% –12%	[139]	ETH Zurich

Table 21: Comparison of selected power amplifiers (PAs) operating in the terahertz band. PAE – power added efficiency defined as:

$$PAE = (P_{OUT\_RF} - P_{IN\_RF}) / P_{DC} \times 100\%.$$

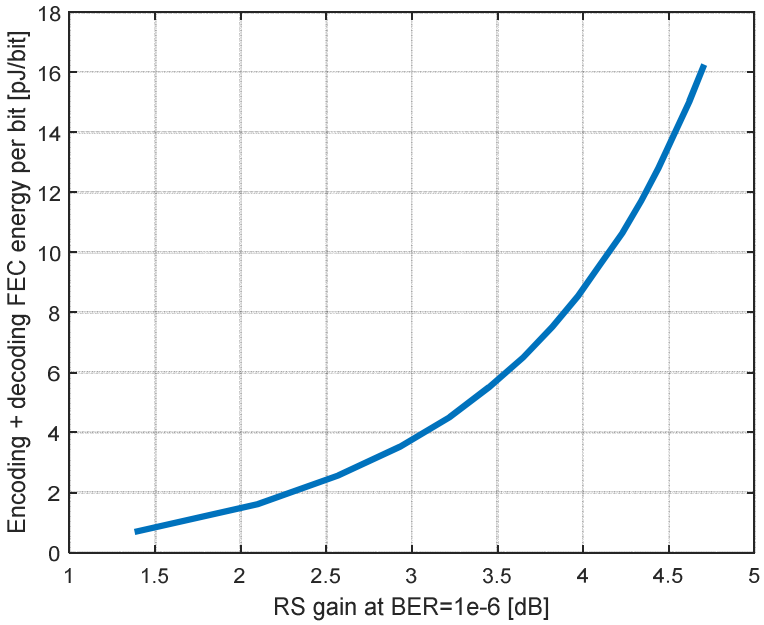


Figure 163: Energy per bit consumed by the data link layer as a function of the RS coding gain (AWGN channel is considered).

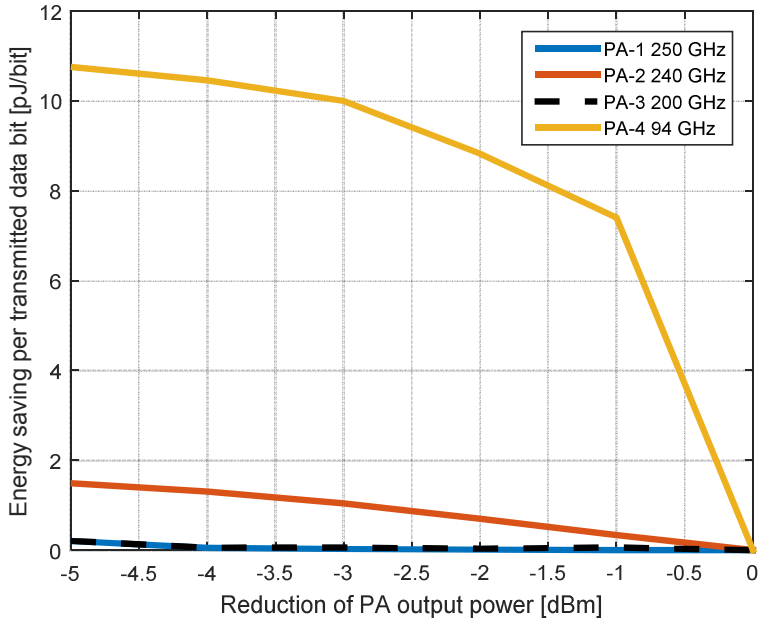


Figure 164: Energy per bit saving when output power of the power amplifiers (PAs) presented in Table 21 is reduced by 5 dBm from the maximal power.

Firstly, 250 GHz power amplifier presented in [137] is considered (denoted as PA-1 in Table 21 and Figure 164). PA-1 achieves up to 10 dBm of output power and efficiency up to ~3%. The authors have published a detailed large-signal performance of the amplifier at 250 GHz. Thus, a precise simulation model can be included in the estimation. Figure 164 shows energy per bit savings when the output power of the considered PAs is reduced. In case of PA-1, Reduction by 4.5 dBm of the output power reduces the consumed DC power by ~0.15 pJ/bit. The RS FEC engine requires ~16.2 pJ/bit to compensate the reduction (an AWGN channel is considered). Thus, the FEC engine requires ~108 times more energy than the power amplifier can save.

The same analysis is performed for RF frontend proposed in [120] (denoted as PA-2 in Table 21 and Figure 164). In this case, PA-2 achieves lower efficiency than PA-1. Despite this, coding provided by the IRS engine is more expensive in terms of consumed energy. Reduction of the output power by 4.5 dBm reduces the consumed energy by ~1.5 pJ/bit (Figure 164). As mentioned before, RS requires ~16.2 pJ/bit to compensate the same value in the data link layer (~11 times more energy). PA-3 has similar characteristic to PA-1 and the FEC engine requires ~100 times more energy than the power amplifier can save.

Additionally, another PA with the following characteristic is considered: P_{out_max} = 21 dBm, maximum efficiency = 12% (at 20 dBm), operational frequency ~94 GHz (denoted as PA-4 in Table 21 and Figure 164). In this case, RS(255,223) coding is ~1.5 times less efficient in terms of consumed energy per bit. However, coding gain obtained by RS(255,253) is ~10 times more efficient than using PA-4 at output power of 21 dBm. Thus, the data link layer should set the PA-4 operating point to 20 dBm and enable RS(255,253) coding instead of using the maximum power of 21 dBm. If more gain is required, then both parameters have to be adopted according to the normalized energy per 1 dB gain.

5. Conclusion

The leading objective of this work was development of concepts and algorithms for parallel processing system with the main focus on improving communication reliability for ultra-high-speed wireless communication. Several aspects regarding the low-level data link layer, i.e., forward error correction (FEC), automatic repeat request (ARQ), and link adaptation at the intended data rate were investigated. In paragraf 3 hardware constraints for the investigated system are identified. Firstly, the required memory resources for the addressed data rate are estimated. This work discovers that the required memory capacity and access time towards the DDR3 memory are massive challenges. The second deeply investigated problem is calculation complexity of Forward Error Correction (FEC) at the targeted speed, which also turned out to be an extremely demanding task. A typical Viterbi decoder with soft decision decoding may occupy a logic area of more than twenty high-end FPGAs. Such implementations are not only too expensive and energy demanding, but also lead to complicated inter-chip networking. To overcome these problems, some popular algorithms were compared in terms of calculation throughput and consumed chip resources. As a result, hard decodable interleaved RS (IRS) codes and improved turbo product codes (TPC) were selected as the base for the forward error correction engine. This work demonstrates a detailed research on all aspects of the dedicated IRS and TPC for 100 Gbps data link layer. The proposed concept uses link adaptation methods concatenated with hybrid automatic repeat request (HARQ), and the concept was successfully validated in the VC709 Virtex7 development platform. The hardware accelerated DLL-engine achieves goodput of ~116 Gbps on a single Virtex7 device, and allows to communicate on links with BER up to ~2e-3. It means, it supports all high-speed 'transceivers' listed in [4]. Furthermore, the designed accelerator operates fully autonomically and transparently for higher layers. These results helped developing a dedicated Matlab simulation tool for the protocol prototyping. The tool finds suitable parameters, and estimates the optimal solution according to the consumed energy and goodput. Moreover, this work presents mathematical models that describe the system. The models reduce simulation time by a factor of ~100. Solving the equations for projected parameters is much faster than empirical simulations, where several MB of data is processed by an emulated wireless channel and FEC objects. This allows to prove practical measurements and prototype new solutions with almost no waiting-time. This work proposes a hardware platform, interconnection, and data processing methodology for 100 Gbps wireless demonstrator. The data link layer hardware accelerator was used to realize low level protocol tasks, which are parallel lanes processing, a dedicated protocol for frame fragmentation and aggregation, a parallel FEC engine, dedicated link adaptation, ACK-frame compression, and a low complexity hybrid-ARQ algorithm. This evaluation setup, based on the new processing approach, is capable to process ~116 Gbps on the state of the art FPGA board. The designed IRS solution uses 96 RS decoders and can correct a burst error with length up to 6912 bits. This

corresponds to ~116 PSSS symbols and to transmission time of ~70 ns at the targeted data rate. To the author's best knowledge, it is the first 100 Gbps data link layer processor dedicated for wireless communication shown in the world (the processor consist of the hardware accelerator and Tilera; concepts and algorithms dedicated for the Tilera processing are not a part of this work).
The most interesting aspect of this work is estimation of the consumed energy per bit for 100 Gbps networks, and a tradeoff between FEC, ARQ, and transmission power in terms of goodput and energy efficiency. Additionally, the work investigates required EbN0 at the PHY layer. The modified TPC methods bring a significant innovation into the work. The TPC code word shown in section 3.16 improves error correction performance, reduces consumed energy per data bit, and enables pipelined processing optimized for hardware decoding. Moreover, 130 and 40 nm ASIC synthesis results of the data link layer hardware accelerator are provided, and the system (data link layer accelerator, baseband, and RF-frontend) is investigated according to consumed energy per bit.

6. Appendix

6.1 Soft and hard decision FEC processing

Soft decodable FEC algorithms (SD-FEC) require a powerful ASIC technology to implement the targeted 100 Gbps system. Even if the processing is perfectly optimized, and all idle cycles are removed from the processing pipeline, most FPGA devices and low cost ASIC technologies (e.g., IHP 130 nm CMOS) cannot support SD-FEC at the targeted data rate. In the architecture investigated in DFG-SPP16155, the GTX/GTH transceivers are proposed for interconnection, and each transceiver delivers 64 data bits in each clock cycle at 156.25 MHz (10 Gbps). Thus, at least ten lanes are required to support the targeted goodput. If a soft decodable LDPC decoder with 4-bit representation is considered (e.g., decoder presented in [75]), then the soft bit values require data buses of size of at least 2560 bits. Interfaces for the SD-FEC require at least four times more logic than for HD-FEC. The FEC processor has to run at 400 Gbps instead of 100 Gbps. Moreover, cables used to connect to the baseband do significant mechanical stress on the PCB. For hard-decision FEC algorithms (HD-FEC), 42 SMA cables are required. With SD-FEC, at least 162 SMA cables and 50 GTH transceivers are necessary. Figure 165 shows the 100 Gbps demonstrator architecture as proposed in DFG-SPP1655. The demonstrator is used to validate concepts and ideas developed by the SP1655 research teams.
Last but not least issue is the effort to perform baseband's ADC conversion. The ADC has to support multibit values and has to be interconnected to the baseband with at least quadruple data rate (100 Gbps × *num_of_soft_bits* = 400 Gbps). For now, the baseband investigated in the Real100.COM project do not have such possibility. Due to these reasons, HD-FEC algorithms are considered for FPGA implementation, and SD-FEC methods can be applied for high performance mixed-signal ASICs only. Use of soft decision methods requires a fully integrated solution that integrates RF-frontend, baseband, ADC, and data link layer in a single chip. This avoids cables and additional serial protocols for interconnection between the modules.

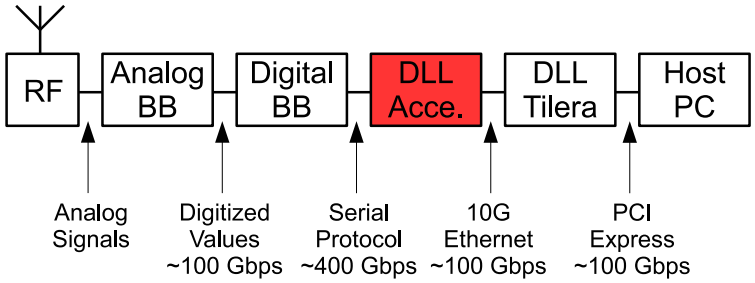


Figure 165: 100 Gbps demonstrator overview with 4 bits SD-FEC. The data link layer hardware accelerator investigated in this work is marked in red.

6.2 Comparison of high-speed serial protocols

The planned 100 Gbps wireless demonstrator consists of several hardware platforms. The RF-frontend and some of the analog BB processing will be realized in a dedicated ASIC. The digital BB will run on an FPGA board. The proposed data link layer accelerator is already successfully validated in VC709 FPGA platform. The higher instance of the data link layer runs in a Tilera many-core processor. All these hardware components need to be connected to each other (Figure 166).

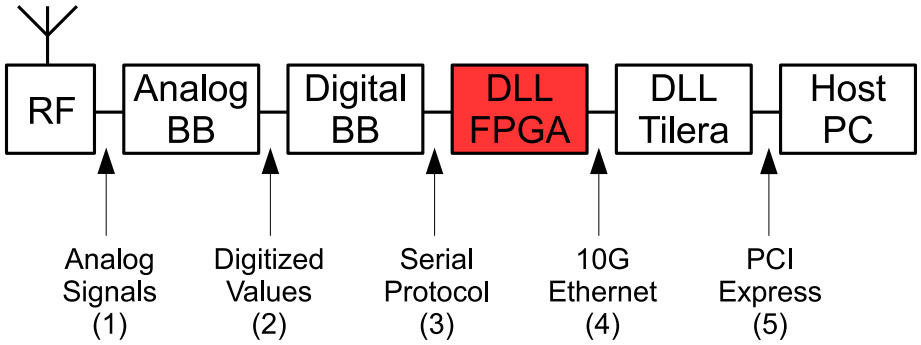


Figure 166: 100 Gbps wireless demonstrator components and planned interconnection between the components. The considered data link layer (DLL) FPGA implementation is marked in red. Most the interconnection is based on high-speed serial protocols.

Some of the interconnection is already defined. For example, the RF-Frontend and Analog BB (position '1' according to Figure 166) have to use analog signals. The interface to the Tilera (4) has to be realized by 10G Ethernet because the Tilera does not support other interfaces. The Tilera and Host PC (5) share common PCI Express lines. The interface on positions (2) and (3) can be adapted according to latency and consumed hardware resources. For this purpose, Aurora protocol [129], Raw GTX [98], RapidIO [140], or 10G Ethernet [101] can be used. All of the protocols use serial links to transfer the data, and all of them are natively supported by mid-range or better FPGA devices. The hardware of the FPGA transceivers can be adjusted to support all the solutions. The difference is latency, achievable goodput, and the cost of the IP-core for ASIC applications. Table 22 compares the most important parameters of the protocols. All measurements are performed on VC709 Virtex7 platform with xc7vx690tffg1761-2 FPGA device.

Parameter	Raw GTX/GTH	Aurora	10G Ethernet	RapidIO gen 2
Latency	18 cycles × 5 ns (90 ns)	47 cycles × 5 ns (235 ns)	53 cyc. × 6.4 ns (340 ns)	110 cyc. x 6.4 ns (700 ns, core ver. 4.0, measured with example provided with Vivado 2015.4)
LUTs	319	2480	2207	6377
FFs	601	1846	2341	7516
Data rate	0.5 – 11.3 Gbps (Virtex7) 28.05 Gbps (UltraScale)	0.75-11.3 Gbps	10.3125 Gbps	1.25 – 6.25 Gbps
Reference clock	9 clock multipliers	17 clock multipliers	156.25 MHz	1 clock multiplayer
Compatibility with other devices	The same FPGA family only (Not standardized)	Standardized	Standardized	Standardized
Flow Control	Emulated via Gearbox	Fully Supported	Emulated by Ethernet commands	Fully Supported
Channel bounding and deskewing	Manual only	Supported	Not supported	Supported
Bus width	32 or 64 bits	64 bits	64 bits	64 bits

Table 22: Comparison of FPGA/ASIC high speed serial protocols for inter-chip communication.

From an efficiency point of view, the latency caused by communication on the serial links increases the overall turnaround time of the system. It means that idle time of the radio increases, because controlling information and data is delayed. This issue can be simulated and influence of the delays on the overall goodput can be estimated. In the targeted system, at least three interfaces use serial interfaces (positions 2, 3, and 4 according to Figure 166). Additionally, the controlling messages have to pass the communication chain four times. Thus, the effective latency seen from the data link layer FSM (Tilera software) is equal to a twelvefold delay of a single period. Therefore, the system performance is affected. Figure 167 compares the system's goodput, when the connections are realized with the mentioned protocols, and compared to an architecture integrated into a single chip. The raw GTX interface reduces the total data rate by ~0.5 Gbps. RapidIO protocol is the least efficient and degrades the goodput by 3.7 Gbps. To achieve the highest efficiency, all demonstrator parts have to be integrated into a single chip, and all controlling signals have to be passed through the structure without any additional delays.
To mitigate the latency caused by FEC and BB processing, the pipeline has to be always filled with data. Therefore, the next data frame has to be prepared for transmission, before the ACK-transaction is finished. In such case, the processing pipeline is always filled with data, and therefore the data processing delay is avoided after processing the ACK.

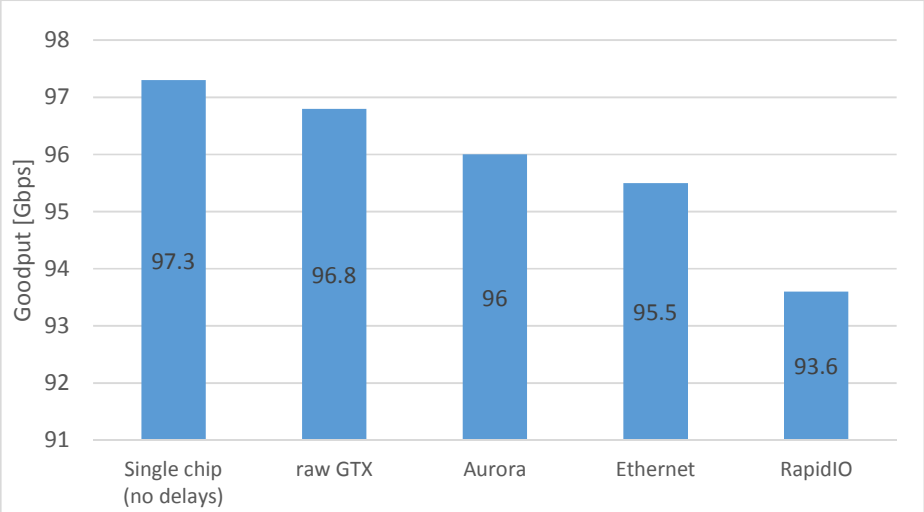


Figure 167: Comparison of the overall system goodput for different FPGA/ASIC high speed serial protocols.

The DLL accelerator uses 10 lanes to process the 100 Gbps stream. Each lane requires two serial links (lane data input and lane data output). Therefore, the DLL FPGA has to be equipped with 20 protocol endpoints. Figure 168 compares consumed FPGA resources required for the interfaces. RapidIO consumes almost 30% of available look-up-tables in the Virtex7 device. The raw GTX protocol is

much more attractive from this point of view and consumes 1.5% LUT resources only.

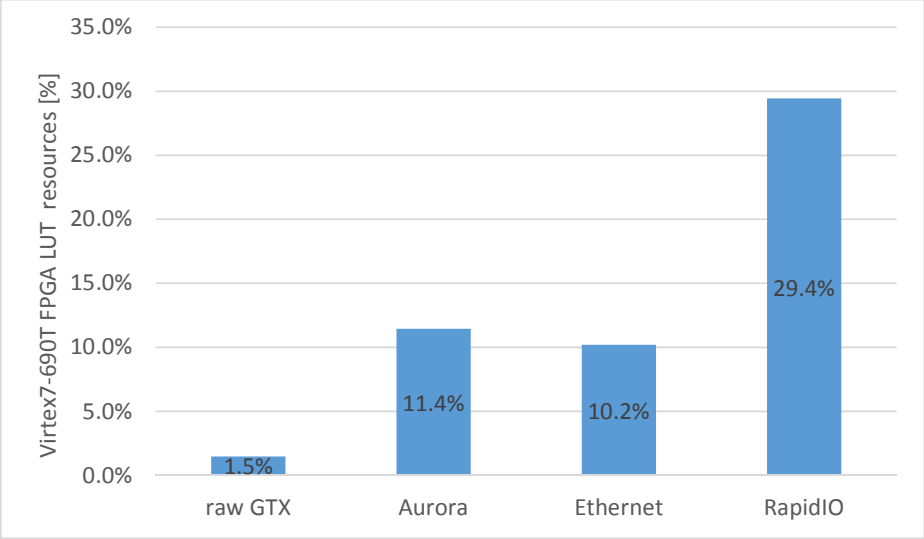


Figure 168: Comparison of consumed FPGA LUT resources required for implementation of 20 serial links required for 100 Gbps operation.

6.3 Lanes deskewing

If several serial transceivers are combined to support high IO throughput, for example when 10×10 Gbps Ethernet lines are combined to achieve 100 Gbps as proposed in DFG-SPP1655, then a skew between the serial transceivers can be observed. It means that data on the Ethernet interfaces is provided for the FPGA/ASIC logic with different phases (Figure 169).

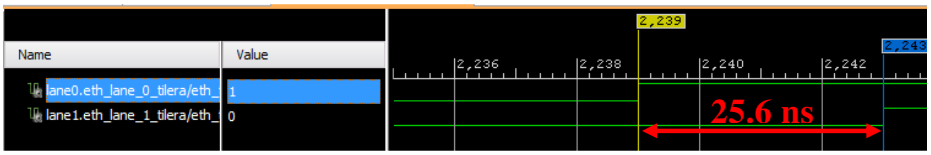


Figure 169: Skew between two 10G Ethernet lanes. The examined transceivers are located side by side in the FPGA chip and thus the skew is relatively small (4 clock cycles at 156.25 MHz). If distant transceivers with different clock sources are selected, the skew is more significant and may cause synchronization issues.

The skew is observed, because signals propagated in the chip, on the PCB, and Ethernet cables have different electrical lengths and are recovered by individual, free-running clock recovery circuits. In some cases, the transceivers may be located in distant FPGA/ASIC parts and may use different clock sources and PLLs [98], [99]. Additionally, each transceiver synchronizes and aligns data words

independently. For this purpose, FIFO buffers are employed in the GTH hardware. During the synchronization, data is stored in the FIFO until the user data clock is synchronized with the recovered data clock. When all the factors cumulate, then the lanes provide the RX data with different phases. There is a possibility to control the synchronization manually, and to adjust the phase skew of all transceiver to the same value. For this purpose, Xilinx proposes a dedicated algorithm to perform the skew alignment for the transceivers (Figure 170).

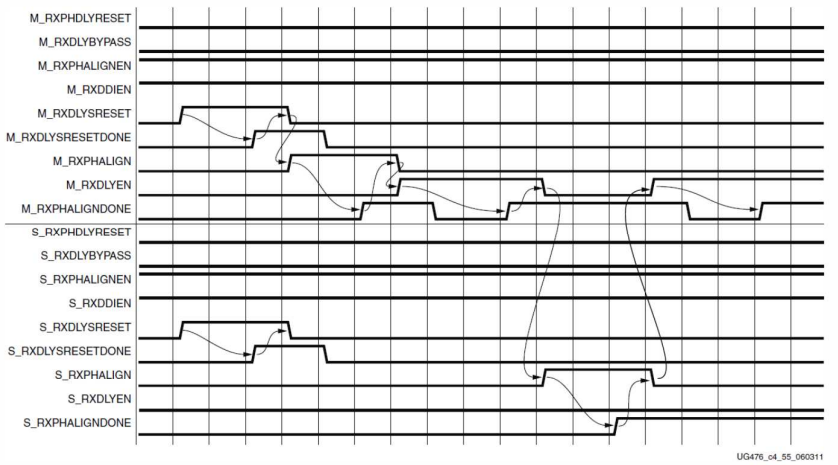


Figure 170: State of the art phase and delay alignment for Virtex7 serial transceivers. Figure retrieved from [98].

In this work, another approach is proposed. The skew alignment can be achieved using dual port FIFO memories like presented in Figure 171.

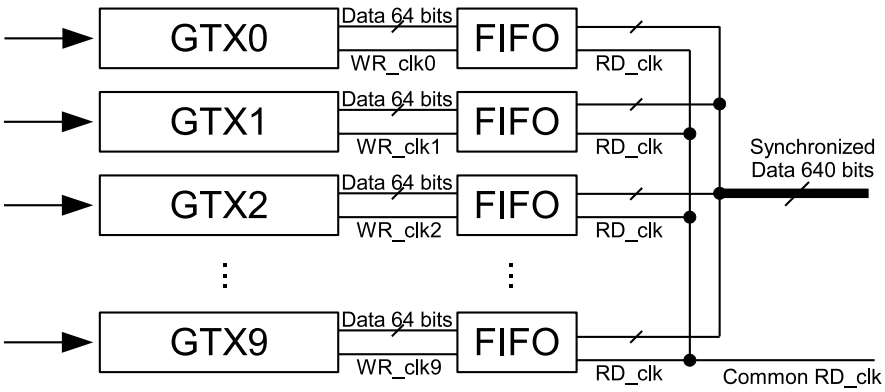


Figure 171: Proposed phase and delay alignment for multiple serial transceivers.

The proposed solution uses dual port FIFO memories to synchronize data streams from the serial transceivers. If at least single element to each FIFO is written, then the logic activates the common RD_clk and receives synchronized data streams from the memories. This solution is simpler to implement and avoids controlling of the signals presented in Figure 170. The overhead caused by additional FIFO buffers is marginal in comparison to the total resources available in nowadays FPGA and ASIC devices.

6.4 On chip flow controlling

From a formal point of view, the proposed FPGA/ASIC processing can be represented as a classical Petri network [141]. The input serial transceivers configured for the Ethernet/GTX/GTH/GTZ operation produce tokens, which are processed by FPGA/ASIC modules. At the end, the tokens are consumed by the output serial transceivers connected to the baseband. Figure 173 shows the implemented Petri network. In the figure, a simplified diagram is shown. The network controls two FEC entities, a single CRC module, and a simplified ACK processing path. In reality, the FPGA uses 96 FEC decoders, 12 CRC decoders, 96 FEC encoders, 12 CRC decoders, 12 aggregation modules, 12 deaggregation modules, clock synchronizers, and several state machines. Thus, the real Petri network is significantly larger than the network presented in Figure 173 (cannot be fitted on a page). Nevertheless, the diagram shows that most the activities are related to the flow control processing. The 100 Gbps operations are relatively complicated due to time constraints, but controlling the data path is causing additional effort. Additionally, the processing pipeline has to work without idle cycles otherwise the average goodput is reduced, and the average power per bit increases.
The designed Petri network allows to test flow control algorithm used for synchronization of the FPGA modules. Processing latency of each transition can be modified, and the pipeline can be checked if the algorithm controls the flow correctly. Moreover, the network is precisely documenting the flow control in a standardized way.
The presented Petri network corresponds to an ACK/REQ protocol used for synchronization [142]. This can be presented as shown in Figure 172.

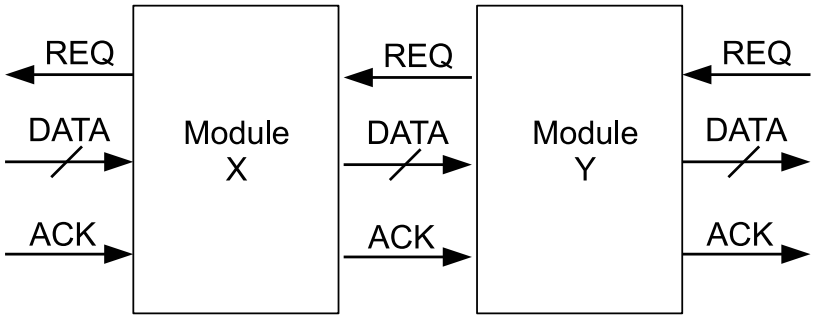


Figure 172: Implemented ACK/REQ protocol for flow controlling of the FPGA pipeline. Figure from data retrieved from [142].

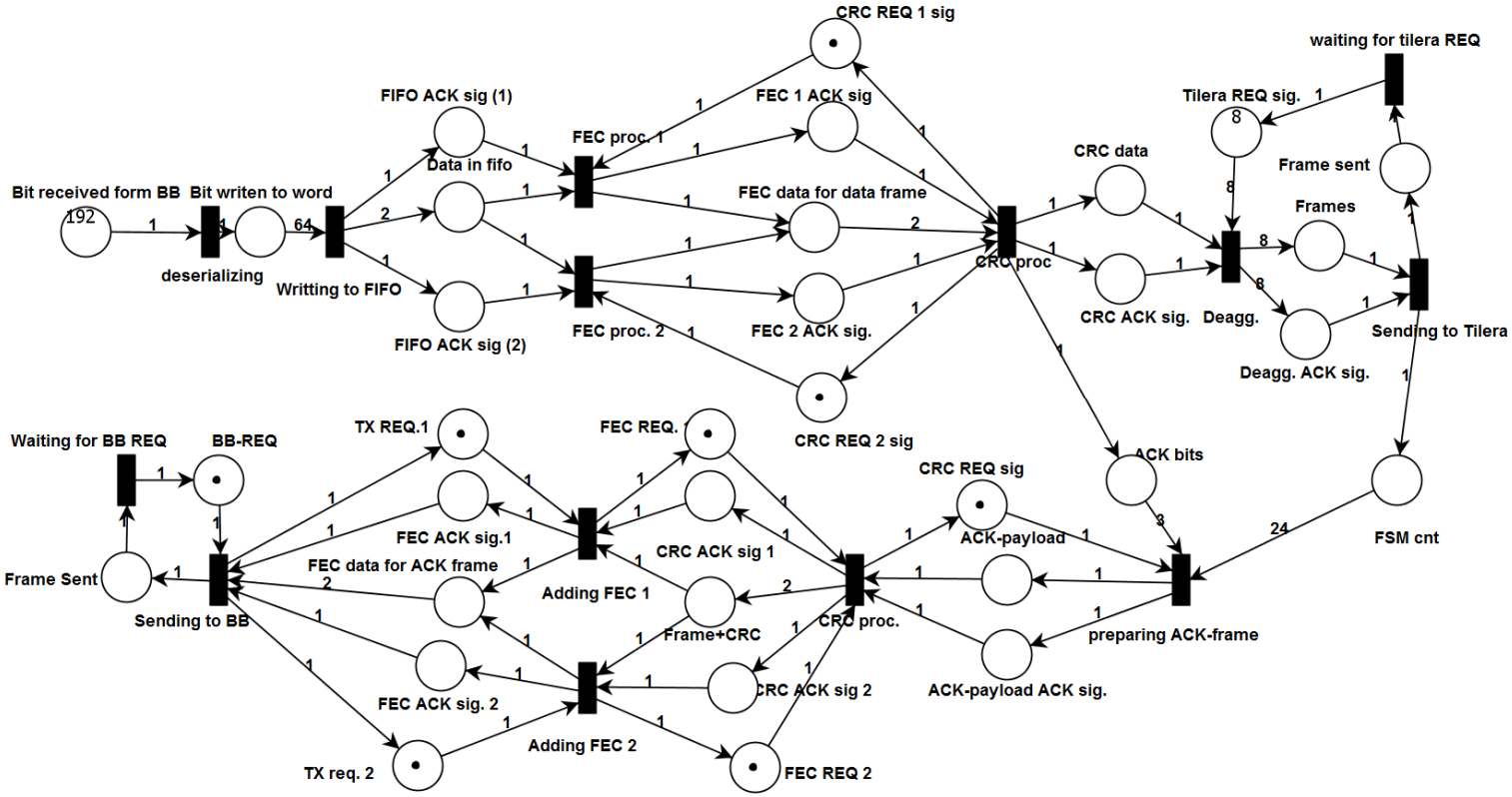


Figure 173: Simplified Petri network for flow controlling of a single processing lane.

6.5 Architecture of a single TX-lane

Figure 174 shows a block diagram of a single transmitter lane. The processing is split into data- and ACK-frame paths. Firstly, data frames received from the Tilera are aggragated. This is necessary due to limited Ethernet encapsulation implemented in the Tilera card. The card can tranmitt Ethernet frames with size up to ~10 kB. However, frames with length of at least 64 kB are required for efficent data tranmission in the targeted wireless system. Thus, the DLL hardware accelerator (FPGA/ASIC) agregates 8 Tilera frames into a single data frame on the fly (the number of aggregated frames is adjustable). After that, the aggregated data is stored in dual port memory. The memory plays a significant role in the processing. It is not only used as a buffer, but additionally synchronizes clock domains between the Ethernet/GTH hardware (156.25 MHz) and data processing logic (~200 MHz in case of Virtex7 FPGA technology). After that, the Tilera header is analyzed and basic information about the frame is extracted from the header (e.g., frame length, ARQ repetition fragment size, etc). This information is necessary to configure FSMs in the CRC and FEC modules. The code rate of the RS encoders is adjustable and can be set to a fixed value, or alternatively can be automatically estimated by the hardware accelerator according to the channel's BER (input BER). After RS encoding, the data is stored in a memory buffer, clock domains are synchronized, and tranmission to the baseband is initialized.
The ACK-frame processing path is a mirror copy of the data processing pipline with some modifications. First of all, the ACK-frame is always encoded with the strongest RS coding available in the system (RS 255,237 for Virtex7 and RS255,223 for 130 and 40 nm CMOS technologies), and there is no link adaptation mechanisms. Additionally, the ACK parser extracts the statistics of the lost frame-fragments from the ACK-frame transmitted by the reciver. This information is used by the channel estimator that controls the RS encoders. From a formal point of view, the channel estimator, RS configuration, and the ACK-frames are a part of a HARQ feedback loop.

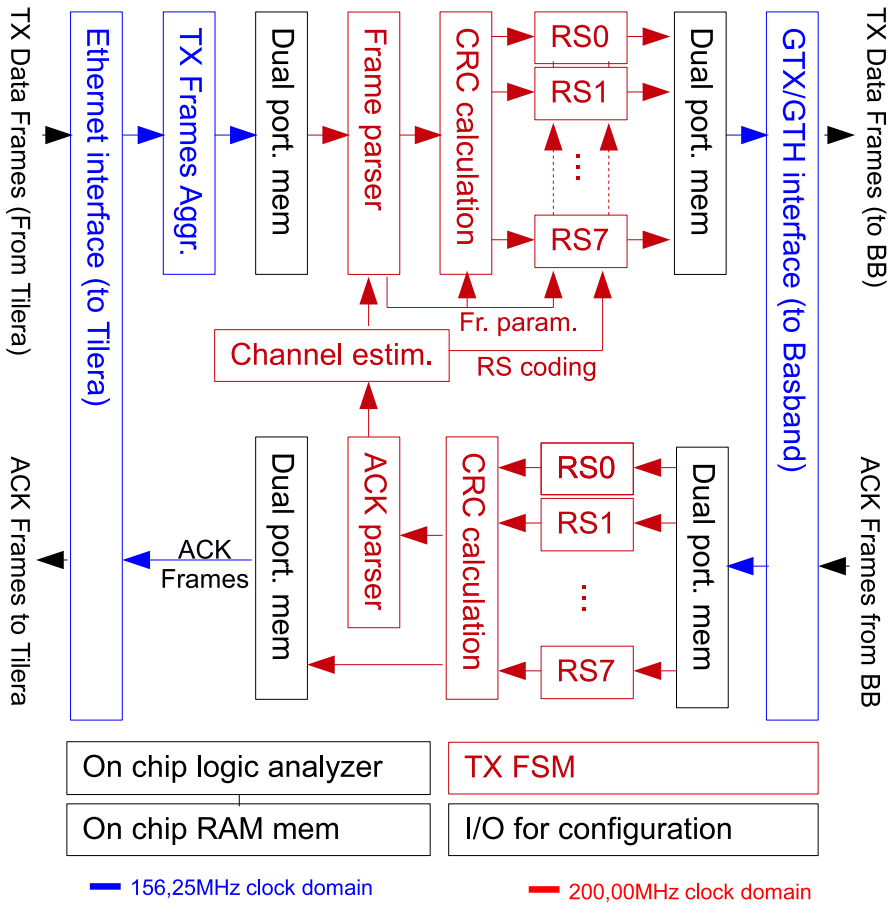


Figure 174: Block diagram of a single TX-lane.

6.6 Architecture of a single RX-lane

The receiver lanes are similar in architecture to the transmitter lanes (Figure 175). The main difference is the ACK-encoder and ACK-generator modules. The ACK encoder compresses the sequence numbers of the delivered frame fragments. Three predefined coding schemes are available (explained in 3.9). The ACK-generator encapsulates the ACK data with a header and all other required information for transmission. Alternatively, the ACK payload can be received from the Tilera processor and the automatic ACK-generation can be disabled in the hardware accelerator.

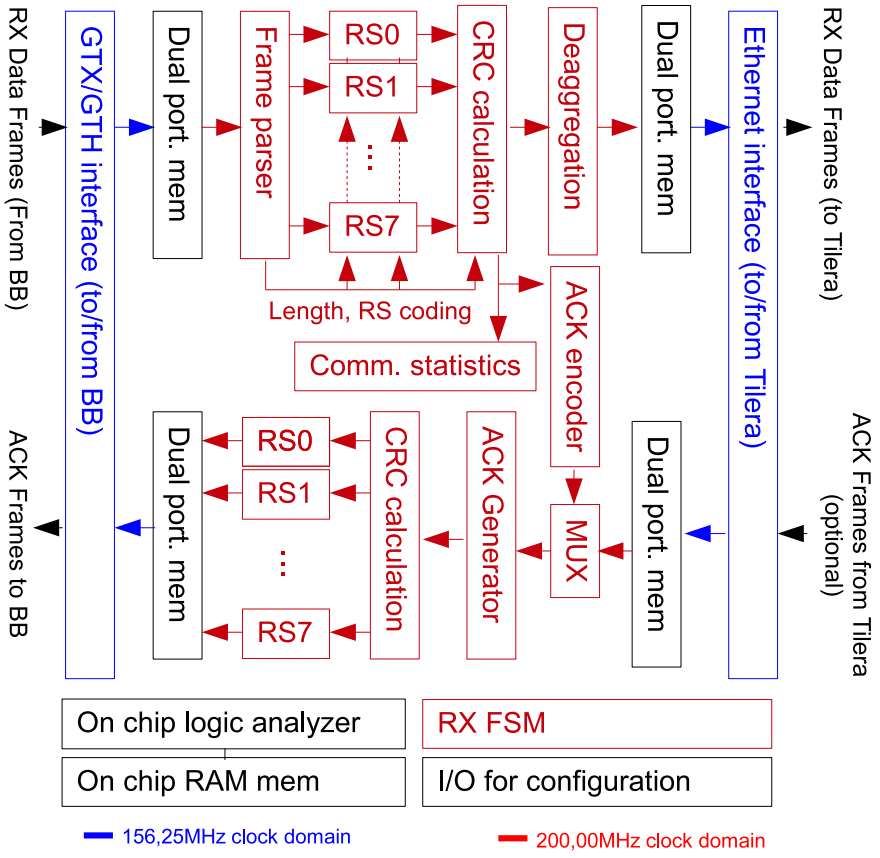
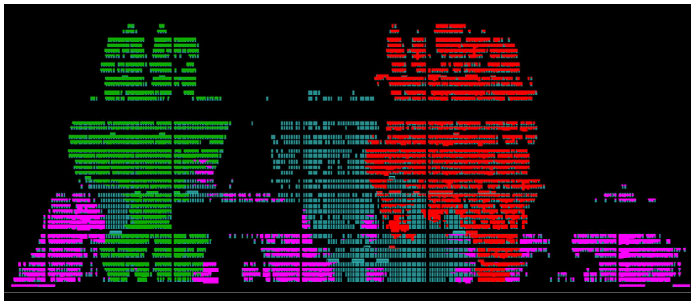


Figure 175: Block diagram of a single RX-lane.

6.7 FPGA floorplan, resources, and clock domains

Figure 176 shows the floorplan of the single RX-lane implemented in the Virtex7 technology. The main parts visible in the floorplan are RS decoders, RS encoders, and Ethernet interfaces with Markov chains. The resources consumed by the mentioned modules are compared in Table 23, Figure 177, and Figure 178. A single RX lane consumes only ~6% of total resources available in the selected FPGA. As expected, the RS encoders and decoders use the largest area of the design. Furthermore, the comparison shows that Ethernet logic adds significant hardware overhead to the design.



- Reed-Solomon encoders
- Reed-Solomon decoders
- Ethernet interfaces
- Other logic

Figure 176: Floorplan of a single RX-lane.

Module	LUTs [pcs]	LUTs [%]	FFs [pcs]	FFs [%]
All Virtex7-690T resources	433200	100%	866400	100%
All consumed resources for 1 lane	25625	5.91%	26062	3.00%
RS decoders	8925	2.06%	7807	0.90%
RS encoders	8148	1.88%	4899	0.56%
Ethernet logic	4858	1.11%	5446	0.62%
RX FSM	1137	0.26%	1013	0.11%
Other logic	2557	0.60%	6897	0.81%

Table 23: Hardware resources consumed by a single RX-lane.

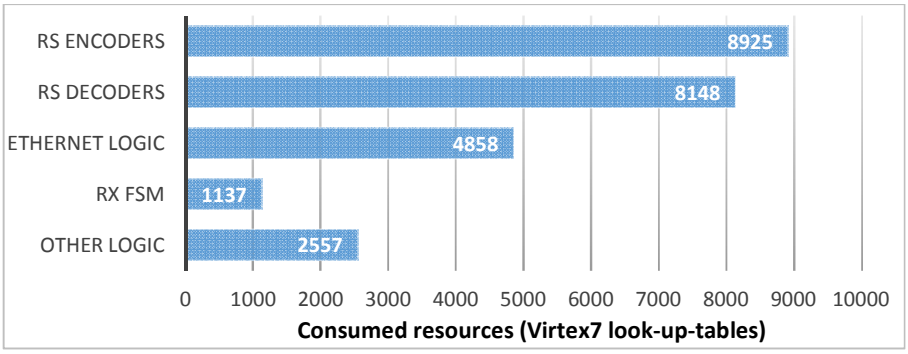


Figure 177: Comparison of hardware resources (look-up-tables) used for a single RX-lane implementation.

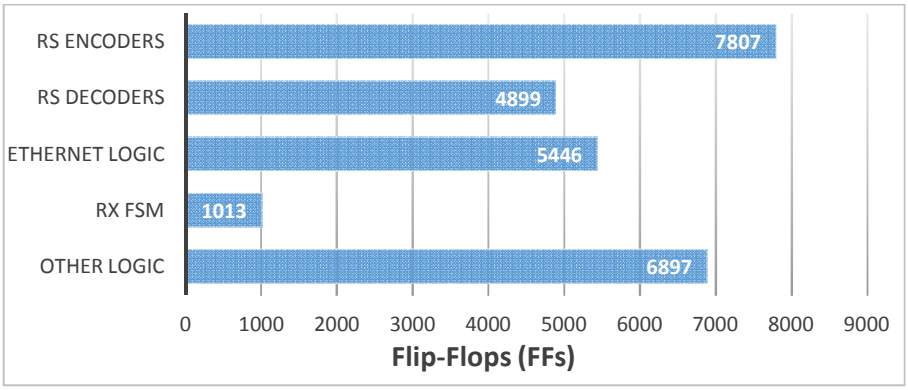


Figure 178: Comparison of hardware resources (flip-flops) used for a single RX-lane implementation (FFs).

Figure 179 shows the FPGA floorplan with four lanes. The lane implementation is split into two parts, which are located around the input and output high-speed Ethernet pins.

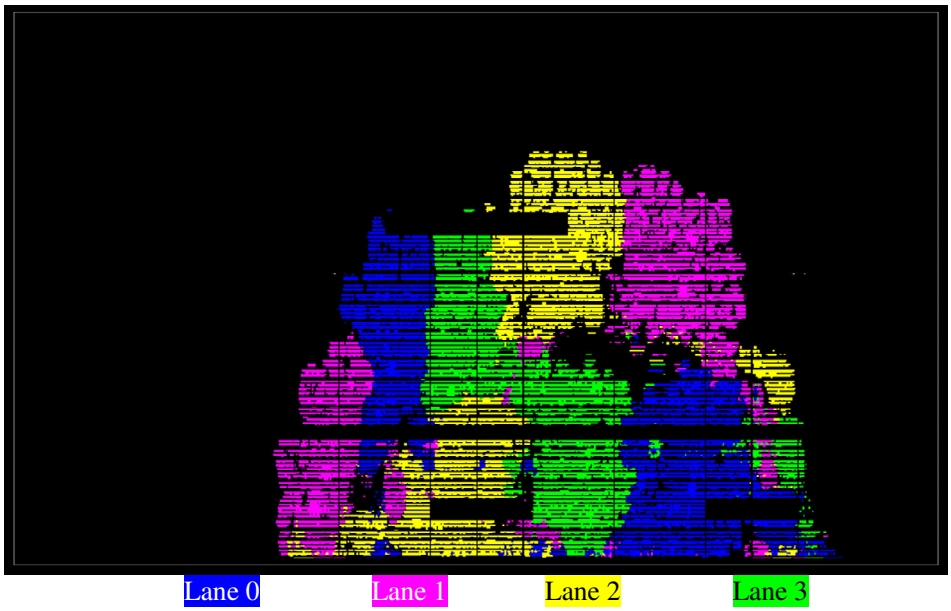


Figure 179: Floorplan of the FPGA with four lanes.

The resources consumed by the implementation can be categorized into three groups: FEC engine (66%), Ethernet communication (19%), and other processes (15%). It means, the FEC is the most resource-hungry operation in the FPGA, while the Ethernet is second. All other modules occupy only 15% of the total resources consumed by the lane (Figure 180).

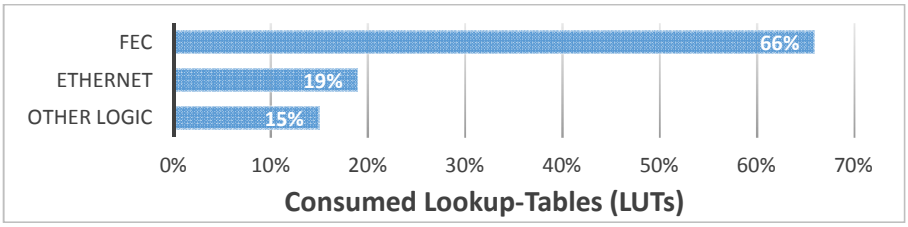


Figure 180: Comparison of hardware resources used for a single RX-lane.

The design uses a relatively large number of clock domains. If a typical configuration with four lanes is considered (according to Figure 148 and Figure 179), then the FPGA uses 21 clocks domains (Figure 181). The DLL processing requires two clocks (156.25 MHz and 200 MHz). One additional clock is required for parameters exchange with a PC (33.33 MHz – jtag clk). One reference clock is used for optical interfaces (156.25 MHz). Furthermore, the FMC card requires a dedicated external reference clock as well (156.25 MHz). Moreover, two clocks domains are generated by each Ethernet transceiver (RX and TX clocks, both 322.27 MHz). Each Ethernet transceiver uses an individual clock recovery circuit. Thus, eight RX-clocks are generated by eight transceivers. Each Ethernet transmitter also uses an individual, hardware generated TX clock. This leads to a complex clock tree, and the design requires a large number of clock synchronization circuits for all signals that cross the clock domains. Thus, timing constraints for the synthesis and implementation are crucial for design stability.

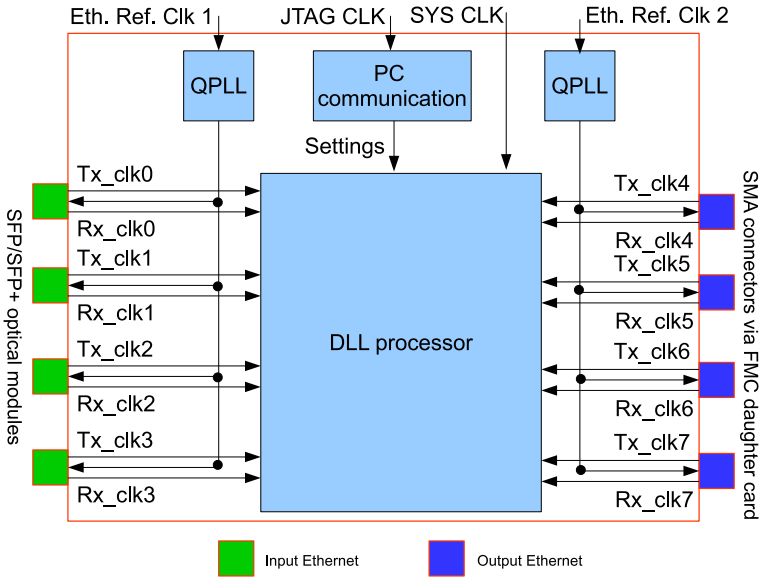


Figure 181: Clock tree of the implemented FPGA accelerator.

6.8 FPGA processing goodput

The FPGA's goodput is strongly correlated with the length of the Tilera input frame. If the frame is longer, the effective goodput of the FPGA accelerator is higher. Due to hardware constraints of the Tilera Ethernet interface, the system cannot use frames longer than ~10 kB. This strongly reduces the transmission goodput of the designed protocol (Figure 182). Additionally, the goodput depends on the RS coding. The RS is a block code, which encodes the data into code words. In case if an incoming frame is indivisible by the RS data block size, the RS code has to be shortened, or the data has to be padded with zeroes. In some cases, code shortening leads to additional idle cycles during decoding [64]. Thus, data padding is a simpler technique, and from the processing efficiency point of view, both techniques are inefficient and should be avoided. To overcome this problem, the FPGA merges the Tilera frames into long data sequences. This is done before the data is forwarded to the FEC engine. Therefore, padding at the end of the code words is reduced. Figure 183 presents results of the proposed algorithm. The solution improves the effective goodput, and solves the issue of short Tilera frames. The goodput of the FPGA processing is defined by the following formula:

$$\eta = \frac{\mathrm{fragLen} * \mathrm{numOfFrags} * \mathrm{aggregFactor}}{\lceil (\mathrm{fragLen}+2) * \mathrm{numOfFrags} * \mathrm{aggregFactor}/\mathrm{rsD} \rceil * rsB + headerL + ackL},$$

where:

fragLen	– HARQ repetition fragment length,
numOfFrags	– number of aggregated HARQ fragments in a frame,
aggregFactor	– number of aggregated Tilera frames,
rsD	– Reed-Solomon data block length,
rsB	– Reed-Solomon code word block length,
headerL	– frame header length,
ackL	– ACK-frame length, in this case it is defined by:

$$\mathrm{ackL} = \lceil (\mathrm{numOfFrags} * \mathrm{numOfFrames} + 2)/\mathrm{rsD} \rceil * rsB + headerL,$$

where:

numOfFrames	– number of frames is a single HARQ session.

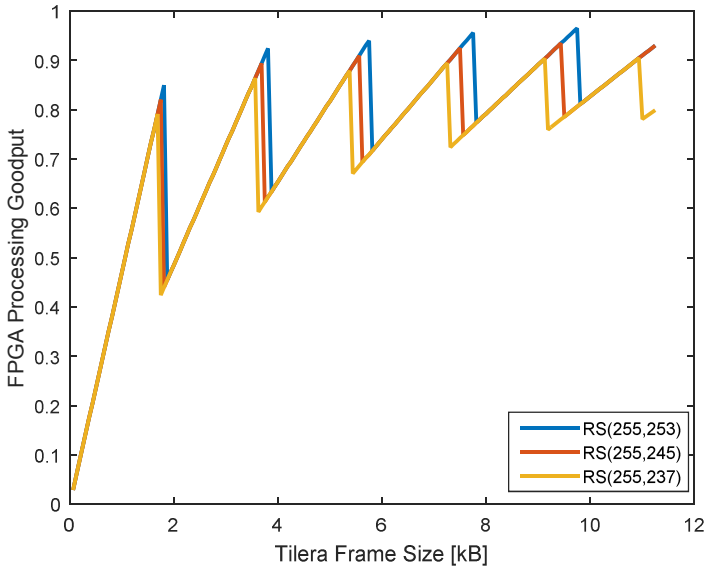


Figure 182: FPGA goodput as a function of the Tilera frame size. The processing goodput of the FPGA is strongly degraded by padding cycles added at the end of the IRS code word. The degradation depends on the size of the Tilera frame.

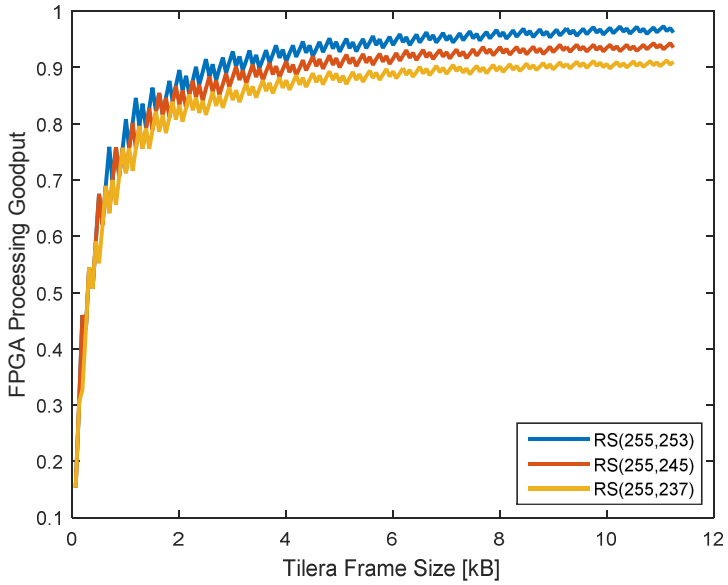


Figure 183: FPGA goodput as a function of the Tilera frame size. The incoming Tilera frames are merged into 64 kB blocks. Therefore, fewer cycles are wasted due to the padding of the RS blocks at the end of frames.

6.9 Consumed energy per bit for accelerator synthesized into 130 nm IHP technology

	RS(255,253)	**RS(255,251)**	**RS(255,249)**	**RS(255,247)**
Energy per bit	*0.75424 [pJ/bit]*	*1.4965 [pJ/bit]*	*2.2269 [pJ/bit]*	*2.9454 [pJ/bit]*
	RS(255,245)	**RS(255,243)**	**RS(255,241)**	**RS(255,239)**
Energy per bit	*3.6519 [pJ/bit]*	*4.3465 [pJ/bit]*	*5.0292 [pJ/bit]*	*5.7 [pJ/bit]*
	RS(255,237)	**RS(255,235)**	**RS(255,233)**	**RS(255,231)**
Energy per bit	*6.3588 [pJ/bit]*	*7.0058 [pJ/bit]*	*7.6407 [pJ/bit]*	*8.2638 [pJ/bit]*
	RS(255,229)	**RS(255,227)**	**RS(255,225)**	**RS(255,223)**
Energy per bit	*8.8749 [pJ/bit]*	*9.4742 [pJ/bit]*	*10.061 [pJ/bit]*	*10.637 [pJ/bit]*
	Frame processing			
Energy per bit	*1.17 [pJ/bit]*			

Table 24: Estimated energy per encoded data bit for intermediate codes (link adaptation, 130 nm IHP technology).

	RS(255,253)	**RS(255,251)**	**RS(255,249)**	**RS(255,247)**
Energy per bit	*1.4761 [pJ/bit]*	*3.724 [pJ/bit]*	*6.028 [pJ/bit]*	*8.3882 [pJ/bit]*
	RS(255,245)	**RS(255,243)**	**RS(255,241)**	**RS(255,239)**
Energy per bit	*10.804 [pJ/bit]*	*13.277 [pJ/bit]*	*15.805 [pJ/bit]*	*18.39 [pJ/bit]*
	RS(255,237)	**RS(255,235)**	**RS(255,233)**	**RS(255,231)**
Energy per bit	*21.031 [pJ/bit]*	*23.728 [pJ/bit]*	*26.481 [pJ/bit]*	*29.29 [pJ/bit]*
	RS(255,229)	**RS(255,227)**	**RS(255,225)**	**RS(255,223)**
Energy per bit	*32.155 [pJ/bit]*	*35.077 [pJ/bit]*	*38.054 [pJ/bit]*	*41.088 [pJ/bit]*
	Frame processing			
Energy per bit	*1.39 [pJ/bit]*			

Table 25: Estimated energy per decoded data bit for intermediate codes (link adaptation, 130 nm IHP technology).

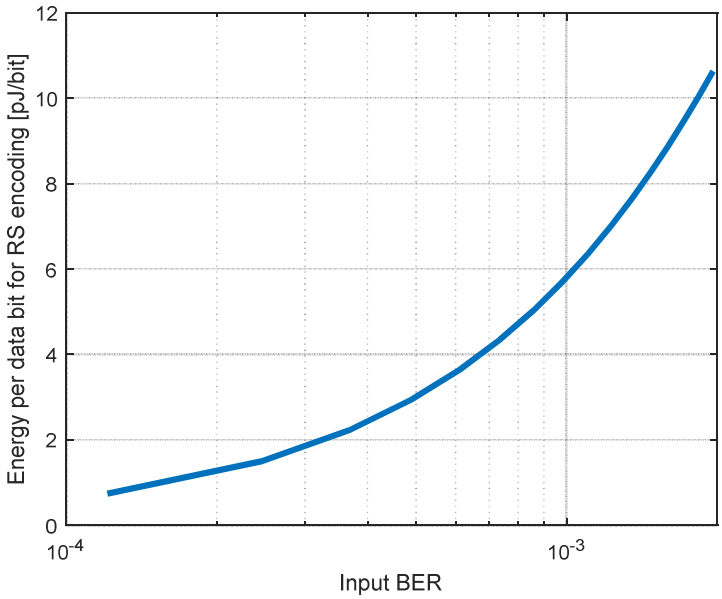


Figure 184: Energy per bit required by the RS encoder as a function of BER (link adaptation, 130 nm IHP technology).

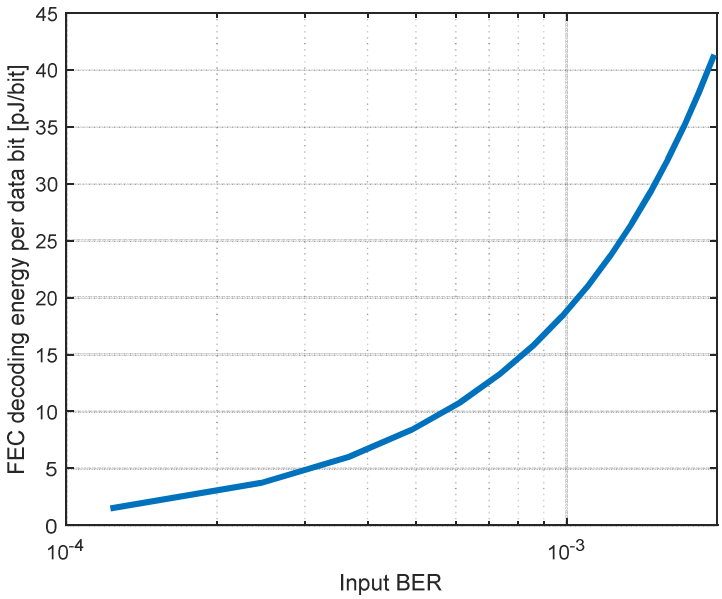


Figure 185: Required energy per processed data bit for RS decoding with enabled link adaptation (130 nm IHP technology). Error characteristic with single errors is used to generate the graph.

6.10 Comparison of IHP RS decoders synthesized into 130 nm IHP technology

	LDPC	***LDPC***	***RS(255,239)***	***RS(255,239)***	***RS(255,223)***
Technology	65 nm GP	65 nm SVT	120 nm Philips	130 nm IHP	130 nm IHP
Code rate	0.5	0.8125 (13/16)	0.9372 (239/255)	0.9372 (239/255)	0.8745 (223/255)
Block size	672	672	2040	2040	2040
Quantization	5 bits	4 bits	1 bit	1 bit	1 bit
Energy Eff.	9 pJ/bit	37 pJ/ bit	6.5 pJ/ bit	18.4 pJ/ bit	38.9 pJ/ bit
Throughput	9 Gbps	161 Gbps	~1.6 Gbps	~2 Gbps	~2 Gbps
Area	1.6 mm^2	12 mm^2	0.16 mm^2	0.21 mm^2	0.38 mm^2
Area Eff. [Gbit/s/mm^2]	5.6	13.6	10	9.5	5.26
Reference	[132]	[75]	[133]	[66]	[66]

Table 26: Comparison of selected LDPC and RS decoders.

6.11 Eb/N0 and energy per bit relation of the accelerator synthesized into 130 IHP technology

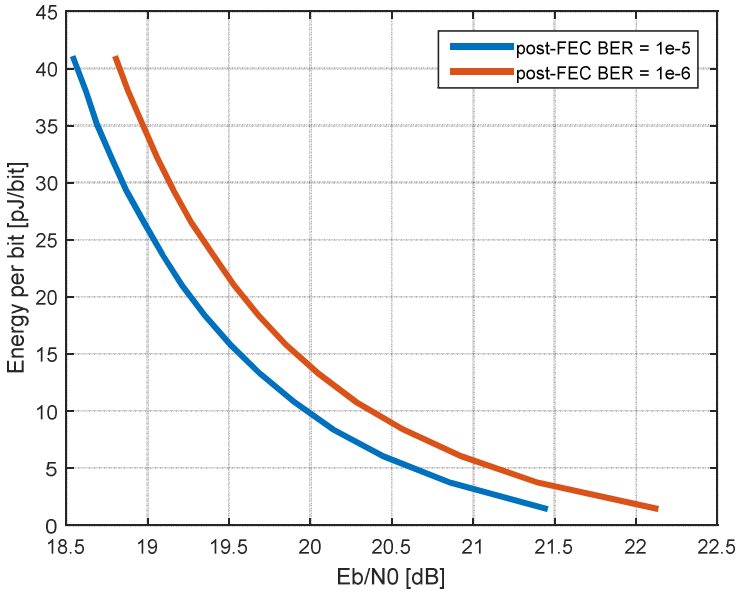


Figure 186: Energy per data bit required by the FEC processor synthesized into 130 nm IHP technology to support the required post-FEC BER: 1e-5 denoted in blue; 1e-6 denoted in red.

6.12 ARQ and FEC tradeoff for accelerator synthesized into 130 nm IHP technology

BER = 4e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,223)	83.678	Optimal mode	1.0451
RS(255,225)	79.84	< 12	1.1052
RS(255,227)	72.467	< 1	1.2284

Table 27: Results of the mathematical model (5.5) for BER = 4e-3 and 130 nm IHP technology.

BER = 3e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1 / (1 - *SER*) (bit retransmission ratio)**
RS(255,223)	87.298	$x_1 \notin R+$	1.0018
RS(255,225)	87.42	Optimal mode	1.0093
RS(255,227)	87.286	< 1770	1.0199
RS(255,229)	86.023	< 302	1.044
RS(255,231)	81.471	< 77	1.1119
RS(255,233)	71.684	< 17	1.2747
RS(255,235)	55.204	< 1	1.6694

Table 28: Results of the mathematical model (5.5) for BER = 3e-3 and 130 nm IHP technology.

BER = 2e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,227)	88.986	$x_1 \notin R+$	1.0004
RS(255,229)	89.634	$x_1 \notin R+$	1.0019
RS(255,231)	90.078	$x_1 \notin R+$	1.0057
RS(255,233)	90.123	Optimal mode	1.0139
RS(255,235)	88.903	< 175	1.0366
RS(255,237)	83.696	< 45	1.1105

Table 29: Results of the mathematical model (5.5) for BER = 2e-3 and 130 nm IHP technology.

BER = 1e-3			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,237)	92.916	$x_1 \notin R+$	1.0003
RS(255,239)	93.394	Optimal mode	1.0035
RS(255,241)	92.912	< 530	1.0172
RS(255,243)	90.051	< 134	1.0582
RS(255,245)	78.285	< 27	1.2273

Table 30: Results of the mathematical model (5.5) for BER = 1e-3 and 130 nm IHP technology.

BER = 1e-4			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,245)	96.078	$x_1 \notin R+$	1
RS(255,247)	96.863	$x_1 \notin R+$	1
RS(255,249)	97.494	Optimal mode	1.0016
RS(255,251)	95.83	< 141	1.0271
RS(255,253)	80.555	< 23	1.2316

Table 31: Results of the mathematical model (5.5) for BER = 1e-4 and 130 nm IHP technology.

BER = 1e-5			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,251)	98.431	$x_1 \notin R+$	1
RS(255,253)	98.604	Optimal mode	1.0062
no coding	79.806	< 5	1.253

Table 32: Results of the mathematical model (5.5) for BER = 1e-5 and 130 nm IHP technology.

BER = 1e-6			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,253)	99.216	Optimal mode	1
no coding	96.94	< 79	1.0316

Table 33: Results of the mathematical model (5.5) for BER = 1e-6 and 130 nm IHP technology.

BER = 1e-7			
FEC	**Goodput [Gbps]**	***x* value [pJ/bit]**	**1/(1-*SER*) (bit retransmission ratio)**
RS(255,253)	99.216	$x_1 \notin R+$	1
no coding	99.388	Optimal mode	1.0062

Table 34: Results of the mathematical model (5.5) for BER = 1e-7 and 130 nm IHP technology.

7. References

[1] L. Lopacinski, M. Brzozowski, R. Kraemer, S. Buechner, and J. Nolte, "100 Gbps wireless – data link layer VHDL implementation," *The 18th Conference on Reconfigurable Ubiquitous Computing, RUC'2015*. 2015.

[2] C. Hermsmeyer, H. Song, R. Schlenk, R. Gemelli, and S. Bunse, "Towards 100G packet processing: Challenges and technologies," *Bell Labs Technical Journal*, vol. 14, no. 2, pp. 57–79, Aug. 2009.

[3] "IEEE 802.3ba-2010: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications." 2010.

[4] G. Ducournau *et al.*, "Ultrawide-Bandwidth Single-Channel 0.4-THz Wireless Link Combining Broadband Quasi-Optic Photomixer and Coherent Detection," *IEEE Transactions on Terahertz Science and Technology*, vol. 4, no. 3, pp. 328–337, May 2014.

[5] G. R. M. Garratt, *The Early History of Radio: From Faraday to Marconi*, vol. 90. 1994.

[6] Texas Instruments, "CC1101 Low-Power Sub-1 GHz RF Transciever." 2015.

[7] IEEE, "Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3 : Enhancements for Very High Throughput in the 60 GHz Band," vol. 2012, no. December. 2012.

[8] J. R. Hopgood and P. J. W. Rayner, "Blind single channel deconvolution using nonstationary signal processing," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 476–488, Sep. 2003.

[9] K. G. Shin, "Goodput analysis and link adaptation for IEEE 802.11a wireless LANs," *IEEE Transactions on Mobile Computing*, vol. 1, no. 4, pp. 278–292, Oct. 2002.

[10] "DFG SPP1655," 2016. [Online]. Available: http://www.wireless100gb.de/. [Accessed: 02-Sep-2016].

[11] A. R. Javed, J. C. Scheytt, K. KrishneGowda, and R. Kraemer, "System design considerations for a PSSS transceiver for 100Gbps wireless communication with emphasis on mixed signal implementation," in *2015 IEEE 16th Annual Wireless and Microwave Technology Conference (WAMICON)*, 2015.

[12] K. KrishneGowda, A. C. Wolf, R. Kraemer, T. Messinger, and I. Kallfass, "Simulation of 100 Gbps using Parallel Sequence Spread Spectrum modulation (PSSS) with 240 GHz radio," in *2015 1st URSI Atlantic Radio Science Conference (URSI AT-RASC)*, 2015.

[13] T. Messinger *et al.*, "Multi-Level 20 Gbit / s PSSS Transmission Using a Linearity-Limited 240 GHz Wireless Frontend," *COMCAS Proc.*, 2015.

[14] A. R. Javed and J. C. Scheytt, "System design and simulation of a PSSS based mixed signal transceiver for a 20 Gbps bandwidth limited communication link," in *2015 1st URSI Atlantic Radio Science Conference (URSI AT-RASC)*, 2015.

[15] H. Schwedick and A. Wolf, "PSSS - parallel sequence spread spectrum a

physical layer for RF communication," in *IEEE International Symposium on Consumer Electronics, 2004*, 2004, pp. 262–265.

[16] S. Koenig *et al.*, "Wireless sub-THz communication system with high data rate," in *Optical Fiber Communication Conference*, 2013.

[17] F. Boes *et al.*, "Ultra-broadband MMIC-based wireless link at 240 GHz enabled by 64GS/s DAC," *2014 39th International Conference on Infrared, Millimeter, and Terahertz waves (IRMMW-THz)*, 2014.

[18] H. Wang *et al.*, "The design, test, and application of the front end in 0.3 THz wireless communication systems," in *Selected Proceedings of the Photoelectronic Technology Committee Conferences held June-July 2015*, 2015.

[19] T. Nagatsuma, K. Kato, and J. Hesler, "Enabling Technologies for Real-time 50-Gbit/s Wireless Transmission at 300 GHz," in *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication - NANOCOM' 15*, 2015.

[20] I. T. Monroy, "Photonic Techniques for Sub-Terahertz Wireless Data Transmission," in *Advanced Photonics 2015*, 2015.

[21] N. Akkari *et al.*, "Joint physical and link layer error control analysis for nanonetworks in the Terahertz band," *Wireless Networks*, 2015.

[22] H.-J. Song, K. Ajito, Y. Muramoto, A. Wakatsuki, T. Nagatsuma, and N. Kukutsu, "24 Gbit/s data transmission in 300 GHz band for future terahertz communications," *Electronics Letters*, vol. 48, no. 15, p. 953, Jul. 2012.

[23] A. Hirata *et al.*, "120-GHz-band millimeter-wave photonic wireless link for 10-Gb/s data transmission," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 5, pp. 1937–1944, May 2006.

[24] X. Pang *et al.*, "100 Gbit/s hybrid optical fiber-wireless link in the W-band (75-110 GHz).," *Optics express*, vol. 19, no. 25, pp. 24944–9, Dec. 2011.

[25] S. Koenig *et al.*, "100 Gbit/s Wireless Link with mm-Wave Photonics," in *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2013*, 2013, p. PDP5B.4.

[26] J. Antes *et al.*, "Transmission of an 8-PSK modulated 30 Gbit/s signal using an MMIC-based 240 GHz wireless link," in *2013 IEEE MTT-S International Microwave Symposium Digest (MTT)*, 2013, pp. 1–3.

[27] J. Antes *et al.*, "220 GHz wireless data transmission experiments up to 30 Gbit/s," in *2012 IEEE/MTT-S International Microwave Symposium Digest*, 2012, pp. 1–3.

[28] H.-J. Song, J.-Y. Kim, K. Ajito, M. Yaita, and N. Kukutsu, "Fully Integrated ASK Receiver MMIC for Terahertz Communications at 300 GHz," *IEEE Transactions on Terahertz Science and Technology*, vol. 3, no. 4, pp. 445–452, Jul. 2013.

[29] T. Nagatsuma *et al.*, "Terahertz wireless communications based on photonics technologies.," *Optics express*, vol. 21, no. 20, pp. 23736–47, Oct. 2013.

[30] S. Koenig *et al.*, "Wireless sub-THz communication system with high data rate," *Nature Photonics*, vol. 7, no. 12, pp. 977–981, Oct. 2013.

[31] X. Li, J. Yu, J. Zhang, Z. Dong, F. Li, and N. Chi, "A 400G optical wireless integration delivery system.," *Optics express*, vol. 21, no. 16, pp. 18812–9,

Aug. 2013.
[32] S. Namiki *et al.*, "Ultrahigh-definition video transmission and extremely green optical networks for future," *IEEE Journal on Selected Topics in Quantum Electronics*, vol. 17, no. 2, pp. 446–457, 2011.
[33] E. Perahia and M. X. Gong, "Gigabit wireless LANs," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 3, p. 23, Nov. 2011.
[34] L. Lopacinski, M. Brzozowski, R. Kraemer, S. Buechner, and J. Nolte, "100 Gbps data link layer – from simulation to FPGA implementation," *Journal of Telecommunications and Information Technology (JTIT)*, 2016.
[35] L. Lopacinski, M. Brzozowski, J. Nolte, S. Buechner, and R. Kraemer, "100 Gbps Wireless - Challenges to the data link layer," *Proc. 2014 IEICE ICTF.*, 2014.
[36] L. Lopacinski, "A 100Gbps Data Link Layer with a Frame Segmentation and Hybrid Automatic Repeat Request," *Science and Information Conference (SAI), 2015*, pp. 1062–1069, 2015.
[37] L. Lopacinski, S. Buechner, and R. Kraemer, "Parallel RS Error Correction Structures Dedicated for 100 Gbps Wireless Data Link Layer," *Ubiquitous Wireless Broadband (ICUWB), 2015 IEEE International Conference on*, 2015.
[38] L. Lopacinski, S. Buechner, and R. Kraemer, "Design and implementation of an adaptive algorithm for hybrid automatic repeat request," *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on*, 2015.
[39] L. Lopacinski, M. Brzozowski, and R. Kraemer, "A 100Gbps data link layer with an adaptive algorithm for forward error correction," *Proceedings of 2015 IEICE ICTF*, 2015.
[40] L. Lopacinski, S. Buechner, and R. Kraemer, "Design and performance measurements of an FPGA accelerator for a 100Gbps wireless data link layer," *Radio Science Conference (URSI AT-RASC), 2015 1st URSI Atlantic*, 2015.
[41] N. Benvenuto and M. Zorzi, *Principles of Communications Networks and Systems*. Chichester, UK: John Wiley & Sons, Ltd, 2011.
[42] R. Jurdak, *Wireless Ad Hoc and Sensor Networks*. Boston, MA: Springer US, 2007.
[43] E. Modiano, "An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols," *Wireless Networks*, vol. 5, pp. 279–286, 1999.
[44] T. Li, Q. Ni, D. Malone, and D. Leith, "Aggregation with fragment retransmission for very high-speed WLANs," *IEEE/ACM Transactions …*, vol. 17, no. 2, pp. 591–604, 2009.
[45] M. Ehrig and M. Petri, "60GHz broadband MAC system design for cable replacement in machine vision applications," *AEU - International Journal of Electronics and Communications*, vol. 67, no. 12, pp. 1118–1128, 2013.
[46] S. Lin and D. Costello, "Error Control Coding: Fundamentals and Applications." 1983.
[47] I. Tinnirello and S. Sunghyun Choi, "Efficiency analysis of burst

transmissions with block ACK in contention-based 802.11e WLANs," in *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, 2005, vol. 5, pp. 3455–3460.
[48] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.
[49] Core Technologies, "Viterbi Algorithm for Decoding of Convolutional Codes." 2014.
[50] Xilinx Corp., "LogiCORE IP Viterbi Decoder v9.0 - Product Guide," in *Xilinx Datasheet*, 2014.
[51] Altera Corp., "Viterbi IP Core User Guide." 2014.
[52] "OpenCores.org." [Online]. Available: http://opencores.org/. [Accessed: 24-Feb-2016].
[53] X. Chen, "Coding in 802.11 WLANs," *National University of Ireland Maynooth*, no. September, 2012.
[54] Institute European Telecommunication Standards, "EN 300 744 - V1.6.1 - Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television," vol. 1, pp. 1–66, 2009.
[55] Creonic, "Viterbi Decoder User Guide," pp. 1–26, 2012.
[56] S. V. Viraktamath, D. G. Talasadar, Girish V. Attimarad, and G.A. Radder, "Performance Analysis of Viterbi Decoder using different Digital Modulation Techniques in AWGN Channel," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, vol. 9, no. 1, pp. 01–06, 2014.
[57] Y. S. Han, P.-N. Chen, Y. S. Han, and P. Chen, "Sequential Decoding of Convolutional Codes," in *Wiley Encyclopedia of Telecommunications*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2003.
[58] Y. S. Han, "BCH Codes Description of BCH Codes."
[59] ETSI, "ETSI EN 302 755 v1.2.1 - DVB; Frame structure channel coding and modulation for a second generation digital terestrial television broadcasting system (DVB-T2)," vol. 1, pp. 1–177, 2010.
[60] "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)," *European Standard (Telecommunications series)*, vol. 1, no. 2, 2009.
[61] R. S. Lim, "A decoding procedure for the Reed-Solomon Codes," *NASA technical paper 1286*, no. August, 1978.
[62] W. Geisel, "Tutorial on Reed-Solomon Error Correction Coding," *NASA Technical Memorandum 102162*, 1990.
[63] Creonic, "100 Gbit/s IEEE 802.3bj RS Encoder and Decoder Product Brief," no. C, pp. 8–11, 2016.
[64] Xilinx Corp., "LogiCORE™ IP Reed-Solomon Decoder Product Guide." 2015.
[65] G. Schmidt, V. R. Sidorenko, and M. Bossert, "Syndrome Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp.

5245–5252, Oct. 2010.
[66] M. Marinkovic, M. Krstic, E. Grass, and M. Piz, "Performance and complexity analysis of channel coding schemes for multi-Gbps wireless communications," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, 2012, pp. 1937–1943.
[67] N. Chen and Z. Yan, "Complexity Analysis of Reed-Solomon Decoding over GF without Using Syndromes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2008, no. 1, p. 843634, 2008.
[68] "Method and apparatus for decoding shortened bch codes or reed-solomon codes," 2008.
[69] B. Yuan, L. Li, and Z. Wang, "Efficient Forward Error Correction Decoder Design for High-Speed Optical Networking," in *Optical Communication*, InTech, 2012.
[70] N. M. El-Gohary, M. A. M. El-Bendary, F. E. A. El-Samie, and M. M. Fouad, "Performance Evaluation of Various Erasures Coding Techniques in Digital Communication," *Journal of Wireless Networking and Communications*, vol. 6, no. 1, pp. 10–20, 2016.
[71] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 5, pp. 641–655, Oct. 2001.
[72] Z. Wang, A. Chini, M. Kilani, and J. Zhou, "Multiple-Symbol Interleaved RS Codes and Two-Pass Decoding Algorithm," *Cornell Univeristy Library*, 2015.
[73] H. Peek, J. Bergmans, J. van Haaren, F. Toolenaar, and S. Stan, *Origins and Successors of the Compact Disc*, vol. 11. Dordrecht: Springer Netherlands, 2009.
[74] V. Guruswami, "Iterative decoding of low-density parity check codes," *Bulletin of the EATCS*, vol. 90, no. 90, pp. 53–88, 2006.
[75] P. Schlaefer, N. Wehn, M. Alles, and T. Lehnigk-Emden, "A New Dimension of Parallelism in Ultra High Throughput LDPC Decoding," *IEEE Workshop on Signal Processing Systems*, pp. 153–158, 2013.
[76] W. Lu, K. Kpalma, and J. Ronsin, "Sparse Binary Matrices of LDPC codes for Compressed Sensing," *Data Compression Conference (DCC)*. p. 10 pages, 10-Apr-2012.
[77] H. Chen, R. G. Maunder, and L. Hanzo, "A survey and tutorial on low-complexity turbo coding techniques and a holistic hybrid arq design example," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1546–1566, 2013.
[78] G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, and W. Ryan, "Design of LDPC codes: A survey and new results," *J. Commun. Softw. Syst*, vol. 2, no. 3, pp. 191–211, 2006.
[79] X. Zhu, A. Doufexi, and T. Kocak, "Throughput and Coverage Performance for IEEE 802.11ad Millimeter-Wave WPANs," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011, pp. 1–5.
[80] T. Brack, M. Alles, F. Kienle, and N. Wehn, "A Synthesizable IP Core for WIMAX 802.16E LDPC Code Decoding," in *2006 IEEE 17th International*

Symposium on Personal, Indoor and Mobile Radio Communications, 2006, pp. 1–5.

[81] F. Naessens *et al.*, "A 10.37 mm2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE," in *2010 Symposium on VLSI Circuits*, 2010, pp. 213–214.

[82] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On Implementation of Min-Sum Algorithm and Its Modifications for Decoding Low-Density Parity-Check (LDPC) Codes," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549–554, Apr. 2005.

[83] Mathworks, "Interleaving." [Online]. Available: http://de.mathworks.com/help/comm/ug/interleaving.html. [Accessed: 26-May-2016].

[84] C. Perkins, *RTP: Audio and Video for the Internet*. Addison-Wesley Professional, 2003.

[85] N. Bhatt, M. Shah, and B. Asodariya, "FPGA Implementation Of Power Efficient Low Latency Viterbi Decoder," vol. 2, no. 5 (May-2013), May 2013.

[86] V. Guruswami and A. Rudra, "Soft decoding, dual bch codes, and better list-decodable e-biased codes," *CCC'08. 23rd Annual IEEE*, 2008.

[87] M. Alekhnovich, "Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, 2002, pp. 439–448.

[88] Intel Corp., "i7-4702MQ." [Online]. Available: http://ark.intel.com/products/75119/Intel-Core-i7-4702MQ-Processor-6M-Cache-up-to-3_20-GHz. [Accessed: 24-Feb-2016].

[89] G. Tzimpragos, C. Kachris, I. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A Survey on FEC Codes for 100G and Beyond Optical Networks," *IEEE Communications Surveys & Tutorials*, 2014.

[90] B. Li, K. Larsen, D. Zibar, and I. T. Monroy, "Over 10 dB net coding gain based on 20% overhead hard decision forward error correction in 100G optical communication systems," *ECOC Technical Digest*, pp. 26–28, 2011.

[91] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *Journal of Lightwave Technology*, vol. 30, no. 1, pp. 110–117, 2012.

[92] D. Arenhage, J. Pettersson, P. Barazandeh, and A. Laszlo, "HSDPA - High Speed Downlink Packet Access Evaluating HARQ with Soft Combining for HSDPA," pp. 1–5.

[93] P. Frenger, S. Parkvall, and E. Dahlman, "Performance comparison of HARQ with Chase combining and incremental redundancy for HSDPA," *IEEE 54th Vehicular Technology Conference. VTC Fall 2001. Proceedings (Cat. No.01CH37211)*, vol. 3, pp. 1829–1833, 2001.

[94] M. W. El Bahri, H. Boujemaa, and M. Siala, "Performance Comparison of Type I, II and III Hybrid ARQ Schemes over AWGN Channels," *Int'l Conf. on Industrial Technology (ICIT)*, pp. 1417–1421, 2004.

[95] I. Joe, "An adaptive hybrid ARQ scheme with concateneted codes for wireless ATM," *MobiCom '97 Proceedings*, pp. 131–138, 1997.

[96] A. Gusmao, R. Dinis, and N. Esteves, "Adaptive HARQ schemes using punctured RS codes for ATM-compatible broadband wireless communications," in *Gateway to 21st Century Communications Village. VTC 1999-Fall. IEEE VTS 50th Vehicular Technology Conference (Cat. No.99CH36324)*, 1999, vol. 5, pp. 2530–2535.
[97] S. Choi and K. Shin, "A class of adaptive hybrid ARQ schemes for wireless links," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 3, pp. 777–790, 2001.
[98] Xilinx Corp., "7 Series FPGAs GTX / GTH Transceivers," *Xilinx Datasheet*. 2012.
[99] "VC709 Evaluation Board for the Virtex-7 FPGA - User Guide," *Xilinx Datasheet*, 2014.
[100] "VC707 Evaluation Board for the Virtex-7 FPGA - User Guide," *Xilinx Datasheet*, 2015.
[101] Xilinx Corp., "10G Ethernet PCS/PMA v6.0," *Xilinx Datasheet*. 2015.
[102] Avago Technologies, "AFBR-709SMZ 10Gb Ethernet, 850 nm, 10GBASE-SR/SW, SFP+ Transceiver." 2013.
[103] Xilinx Corp., "UltraScale Architecture and Product Overview." 2016.
[104] Xilinx Corp., "All Programmable 7 Series - Product Tables and Product Selection Guide." .
[105] Tilera Corp., "Tilera Gx-72 Processor," 2016. [Online]. Available: http://www.mellanox.com/page/products_dyn?product_family=238. [Accessed: 24-Feb-2016].
[106] IBM Corp., "IBM Bladecenter," 2016. [Online]. Available: http://www-05.ibm.com/de/bladecenter/. [Accessed: 02-Feb-2016].
[107] M. Marinkovic, M. Piz, G. Panic, M. Ehrig, and E. Grass, "Performance evaluation of channel coding for Gbps 60-GHz OFDM-based wireless communications," *IEEE Int. Symp. on Personal, Indoor and Mobile Radio Comm. (PIMRC)*, pp. 994–998, 2010.
[108] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas, "LTE-advanced: next-generation wireless broadband technology," *IEEE Wireless Communications*, vol. 17, no. 3, pp. 10–22, Jun. 2010.
[109] L. Badia, N. Baldo, M. Levorato, and M. Zorzi, "A Markov framework for error control techniques based on selective retransmission in video transmission over wireless channels," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 488–500, Apr. 2010.
[110] "Enhancements to 802.3 media access control and associated signaling schemes for ethernet switching." 30-Sep-1997.
[111] Altera Corp., "40- and 100-Gbps Ethernet MAC and PHY MegaCore Function User Guide." 2014.
[112] N. Ansari, "TCP in wireless environments: problems and solutions," *IEEE Communications Magazine*, vol. 43, no. 3, pp. S27–S32, Mar. 2005.
[113] J. Luo, A. Kortke, and W. Keusgen, "An Efficient Frame Aggregation and Block-ACK Scheme for 60 GHz Short-Range Point-to-Point Transmission," *Wireless Personal Communications*, 2012.
[114] J. Il Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, "Achieving single

channel, full duplex wireless communication," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking - MobiCom '10*, 2010, p. 1.
[115] M. Duarte, "Full-duplex Wireless: Design, Implementation and Characterization," 2012.
[116] Xilinx Corp., "VCU108 Evaluation Board - User Guide." 2015.
[117] InveaTech, "CESNET and INVEA-TECH demonstrate 100 Gbps transfers over PCIe with a single FPGA." 2015.
[118] P. Reviriego, B. Huiszoon, V. López, R. B. Coenen, J. A. Hernández, and J. A. Maestro, "Improving Energy Efficiency in IEEE 802.3ba High-Rate Ethernet Optical Links," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 17, no. 2, pp. 419–427, 2011.
[119] K. Krishnegowda, A. Wolf, R. Kraemer, J. C. Scheytt, and I. Kallfass, "Wireless 100 Gb/s: PHY layer overview and challenges in the THz freqency band," *2014 IEEE 15th Annual IEEE Wireless and Microwave Technology Conference, WAMICON 2014*, 2014.
[120] N. Sarmah *et al.*, "A fully integrated 240-GHz direct-conversion quadrature transmitter and receiver chipset in SiGe technology," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 2, pp. 562–574, 2016.
[121] M. Marinkovic, M. Krstic, E. Grass, and M. Piz, "Performance and complexity analysis of channel coding schemes for multi-Gbps wireless communications BT - Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on," pp. 1937–1943, 2012.
[122] M. A. Ingale, "Error Correcting Codes in Optical Communication Systems," no. January, 2003.
[123] Xilinx Corp., "LogiCORE™ IP Reed-Solomon Encoder Product Guide." 2015.
[124] Ji and Michael, "An optimized processor for fast Reed-Solomon encoding and decoding," *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, p. III-3097-III-3100, 2002.
[125] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, p. 3, Jan. 2001.
[126] ITU Recommendation, "Itu-T G.709/Y.1331," 2012.
[127] ITU Recommendation, "ITU-T G.975," vol. 975, 2000.
[128] ITU Recommendation, "G.975.1 (02/2004)," 2004.
[129] Xilinx, "LogiCORE IP Aurora 64B/66B v9.2 - Product Guide." 2014.
[130] B. Steffen, L. Lopacinski, J. Nolte, and R. Kraemer, "100 Gbit/s End-to-End Communication: Designing Scalable Protocols with Soft Real-Time Stream Processing," *in Proc. IEEE Local Computer Networks Conference*.
[131] Xilinx, "I/O Design Flexibility with the FPGA Mezzanine Card (FMC)." 2009.
[132] Y. S. Park, D. Blaauw, D. Sylvester, and Z. Zhang, "A 1.6-mm^2 38-mW 1.5-Gb/s LDPC decoder enabled by refresh-free embedded DRAM," in *2012 Symposium on VLSI Circuits (VLSIC)*, 2012, pp. 114–115.

[133] A. Kumar and S. Sawitzki, "High-Throughput and Low-Power Architectures for Reed Solomon Decoder," *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.*, pp. 990–994, 2005.
[134] J. Hong and M. Vetterli, "Simple algorithms for BCH decoding," *IEEE Transactions on Communications*, vol. 43, no. 8, pp. 2324–2333, 1995.
[135] Texas Instruments, "WL18x7MOD WiLink 8 Dual-Band Industrial Module – Wi-Fi, Bluetooth, and Bluetooth Low Energy (LE)." 2014.
[136] Skyworks, "SKY85803: Dual-Band 802.11a/b/g/n/ac WLAN Front-End Module." 2015.
[137] S. Diebold *et al.*, "A novel 1 - 4 coupler for compact and high-gain power amplifier MMICs around 250 GHz," *IEEE Transactions on Microwave Theory and Techniques*, vol. 63, no. 3, pp. 999–1006, 2015.
[138] I. Kallfass *et al.*, "A 200 GHz monolithic integrated power amplifier in metamorphic HEMT technology," *IEEE Microwave and Wireless Components Letters*, vol. 19, no. 6, pp. 410–412, 2009.
[139] D. Marti *et al.*, "94-GHz large-signal operation of AlInN/GaN high-electron-mobility transistors on silicon with regrown ohmic contacts," *IEEE Electron Device Letters*, vol. 36, no. 1, pp. 17–19, 2015.
[140] S. Fuller, *RapidIO: The Embedded System Interconnect*. 2005.
[141] A. Tzes and W. R. McShane, "Applications of Petri networks to transportation network modeling," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 2, pp. 391–400, May 1996.
[142] Xilinx Corp., "AXI Reference Guide." 2011.
[143] I. Bergel and S. Benedetto, "The effective coherence time of common channel models," in *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, 2010.